User's Guide

AcuBench®

Version 8.1.3

Micro Focus
9920 Pacific Heights Blvd
San Diego, CA 92121
858.795.1900

E-01-UG-100501-AcuBench-8.1.3

Contents

Chapter 1: Introduction

1.1 Product Overview 1-2
1.2 Companion Products 1-4
1.3 Technical Services 1-6

Chapter 2: Getting Started

2.1 Introduction 2-2
2.2 About AcuBench Documentation 2-2
2.3 Related Documents 2-4
2.4 Notation 2-5
2.5 Sample Programs 2-5
2.6 System Requirements 2-6
2.7 Installing and Uninstalling AcuBench 2-6
2.8 Navigating the User Interface 2-7
  2.8.1 The Toolbars 2-9
  2.8.2 The Workspace Window 2-11
  2.8.3 The Development Window 2-12
  2.8.4 The Screen and Report Component Toolboxes 2-12
  2.8.5 The Output Window 2-13
  2.8.6 The Status Bar 2-13
2.9 Printing in the Workbench 2-14

Chapter 3: Workbench Concepts

3.1 AcuBench Concepts 3-2
3.2 Project Management, Organization, and Structure 3-2
  3.2.1 The Workspace 3-3
  3.2.2 Projects 3-5
  3.2.3 Programs 3-6
3.3 Automatic Code Generation 3-7
  3.3.1 Generated COPY Files 3-8
  3.3.2 Controlling code generation 3-9
3.4 AcuBench File Types 3-10
3.5 Development Approaches 3-12
  3.5.1 General Considerations 3-13
    3.5.1.1 Project prefix settings 3-14

3.5.1.2 Project directories....3-14
3.5.1.3 Placing files in the project directory....3-15
3.5.1.4 Keyboard shortcut settings....3-15
3.5.1.5 Setting options and testing....3-16
3.5.2 Developing New Applications....3-17
3.5.3 Developing Programs That Do Not Use Code Generating Tools....3-19
3.5.4 Working with Large Applications....3-20
3.5.4.1 Factors that affect performance....3-20
3.5.4.2 Organizing your workspace....3-21
3.5.4.3 The command-line interface to AcuBench....3-22
3.6 Using Thin Client Technology with AcuConnect....3-24

Chapter 4: Customize Your Working Environment

4.1 Introduction....4-2
4.2 The Tools/Options Dialog....4-2
4.3 Setting Environment Options....4-5
4.3.1 General Environment Options....4-5
4.3.2 Template Options....4-7
4.3.3 Version Control Options....4-8
4.3.4 Build Options....4-9
4.3.5 Debug Options....4-10
4.3.6 Keyboard Options....4-11
4.3.7 Prefix Options....4-12
4.3.8 Miscellaneous Environment Options....4-13
4.4 Setting Code Editor Options....4-15
4.4.1 Code Editor General, Format, Tabs, and Keyword Options....4-15
4.4.2 Code Insight Options....4-16
4.4.3 Paragraph List, Variable List, Constant List, and COPY File List Options.4-17
4.5 Setting Screen Designer Options....4-19
4.6 Setting Code Generator Options....4-20
4.6.1 Generate Document Options....4-20
4.6.2 Program Tag Options....4-22
4.7 Setting Data Designer Options....4-28
4.8 Setting Report Writer Options....4-29
4.9 The Customize Dialog....4-30
4.9.1 Customizing AcuBench Toolbars....4-31
4.9.2 Accessing External Applications....4-33

Chapter 5: Version Control

5.1 Version Control Overview 5-2
5.1.1 Working with version control in AcuBench 5-3
5.1.2 Preliminary considerations 5-3
5.2 The Version Control Interface 5-5
5.2.1 Displaying the Version Control Window 5-5
5.2.2 The Version Control Interface Fields 5-6
5.3 Adding Commands to the Menu Command List 5-8
5.4 Modifying the Command List 5-8
5.5 Saving the Command List 5-9
5.6 Command Variables 5-10
5.7 Issuing Version Control Commands 5-11

Chapter 6: Working with Projects

6.1 AcuBench Project Management 6-2
6.2 Working at the Workspace Level 6-3
6.2.1 Creating a Workspace 6-3
6.2.2 Opening a Workspace 6-4
6.2.3 Saving a Workspace 6-4
6.2.4 Building a Workspace 6-4
6.2.5 Rebuilding a Workspace 6-5
6.2.6 Regenerating a Workspace 6-6
6.2.7 Stopping a Build 6-6
6.2.8 Closing a Workspace 6-6
6.3 Working with Projects 6-7
6.3.1 Creating a Project 6-7
6.3.2 Project Properties 6-10
6.3.3 Adding Folders to a Project 6-12
6.3.4 Adding Files to a Project 6-13
6.3.5 Removing Files From a Project 6-15
6.3.6 Moving Components Among Projects and Folders 6-16
Moving a Program, Screen, Report, or Data Layout to Another Project 6-16
Moving a File to Another Folder or Project 6-16
6.3.7 Deleting a Project 6-16

Chapter 7: Project Settings

7.1 Introduction....7-2
7.2 Modes....7-3
7.2.1 Working with Modes....7-4
7.2.2 Adding a Mode....7-4
7.2.3 Removing a Mode....7-5
7.2.4 Saving a Mode as a POF....7-6
7.2.5 Reusing a POF....7-6
7.2.6 Switching Between Modes....7-7
7.3 Compiler and Runtime Settings....7-7
7.4 Working with Runtime Configuration Files....7-9
7.4.1 Editing Configuration Files....7-10
7.4.2 User-defined Variables....7-12
7.5 Environment Settings....7-12
7.5.1 Working with Environment Variables....7-14
7.5.2 The COPYPATH Environment Variable....7-14
7.6 Library Settings....7-16

Chapter 8: Working with Data at the Project Level

8.1 Introduction....8-2
8.2 Defining Data Files for Use in Projects and Programs....8-2
8.3 Creating Data Layout Files....8-4
8.3.1 Creating a DLT from Scratch....8-5
8.3.2 Creating a DLT from a Single FD/SL Pair....8-6
8.3.3 Creating DLTs from Multiple FD/SL Pairs....8-8
8.4 Working in the File Designer....8-10
8.4.1 Adding File Control Information....8-10
8.4.2 Adding a File Description....8-13
8.4.2.1 Linking and importing COPY files....8-17
8.4.2.2 Occurs Syntax dialog....8-18
8.4.2.3 The Field dialog....8-19
8.4.3 Defining Key Information....8-20
8.4.4 Defining File Handling Behavior....8-22
8.4.5 Designing a Custom XFD....8-24
8.5 Copying DLT Files Between Projects....8-28
8.6 Tips for Working in the Data View....8-29
8.6.1 Useful Data View Functions....8-30
8.6.2 Data Layout Properties....8-31

Chapter 9: Working with Programs

9.1 Introduction....9-2
9.2 Creating an AcuBench Program....9-2
9.2.1 Adding an AcuBench Program to a Project....9-4
9.2.2 AcuBench Program Properties....9-5
9.3 Adding and Creating Basic Source Files....9-12
9.3.1 Creating a File....9-12
9.3.2 Adding an Existing Source File....9-14
9.3.3 Adding COPY Files....9-16
9.3.4 Working with Files in the File View....9-18
9.3.5 File Properties....9-19
9.3.6 Reparsing Source Files....9-20
9.4 Generating a Program....9-21
9.5 Compiling a Program....9-21
9.5.1 Compiling Programs to a Server....9-22
9.5.2 Compiling Multiple Programs....9-24
9.5.3 Remote Precompiling with Boomerang....9-24
9.6 Executing a Local Program....9-26
9.7 Executing a Remote Program....9-26
9.8 Debugging a Program....9-27
9.8.1 Debugging with the Thin Client....9-28
9.8.2 Debugging a Transaction Processing (TP) Application....9-28

Chapter 10: Working with Data at the Program Level

10.1 Introduction....10-2
10.2 Using the Data Set Designer....10-2
10.2.1 Creating a Data Set....10-3
10.2.2 Opening an Existing Data Set....10-7
10.2.3 Creating a BEFORE or AFTER Procedure....10-7
10.2.4 The Data Set's Property Window....10-9
10.2.5 Generating File Handling Code....10-9
10.3 Using the Working-Storage and Linkage Editors....10-11
10.3.1 The Working-Storage Editor....10-11
10.3.2 The Linkage Editor....10-12
10.3.3 The Working-Storage and Linkage Editor Interface....10-12

Chapter 11: Configuring the Code Editor

11.1 Introduction 11-2
11.2 Establishing Keyboard Shortcuts 11-2
11.3 Customizing the Code Editor Interface 11-4
    11.3.1 Configuring Basic Editor Functions 11-4
    11.3.2 Modifying Editor Appearance 11-5
    11.3.3 Customizing Tab Stops 11-7
    11.3.4 Configuring Keyword Behaviors 11-8
11.4 Setting Print Layout Options 11-10
    11.4.1 Setting Headers and Footers 11-11
    11.4.2 Setting Page Appearance 11-11

Chapter 12: Working with Source Code

12.1 Introduction 12-2
12.2 Working with Files 12-2
    12.2.1 Creating a New File 12-3
    12.2.2 File Formats 12-4
12.3 The Code Editor Window 12-4
12.4 Basic Editor Functions 12-5
    12.4.1 Using Bookmarks 12-7
    12.4.2 Changing Case and Indenting Lines 12-8
    12.4.3 Viewing Multiple Sections of Your File 12-8
    12.4.4 Merging Data From Another File 12-9
12.5 COBOL-Friendly Editing Functions 12-9
    12.5.1 Adding and Removing Line Numbers 12-9
    12.5.2 Working with Tabs 12-10
    12.5.3 Working with Blocks of Code 12-11
    12.5.4 Using Code Insight Functions 12-12
    12.5.5 Using Paragraph, Variable, and Constant Lists 12-14
    12.5.6 Working with COPY Files 12-16
    12.5.7 Using Source Code Templates 12-17
    12.5.8 Navigating between error lines 12-17

Chapter 13: Configuring the Screen Designer

13.1 Introduction 13-2
13.2 Customizing the Screen Designer Interface 13-2

13.3 Establishing Screen and Control Defaults 13-4
13.4 Adding Screen Templates 13-6
13.5 Configuring Keyboard Shortcuts 13-7

Chapter 14: Working with Screens

14.1 Introduction 14-2
14.2 Creating a New Screen 14-3
14.3 Getting Started with Screen Design 14-5
14.3.1 Setting Basic Screen Form Properties 14-6
14.3.2 Creating a Resizable Screen 14-8
14.4 Adding Controls to a Screen 14-9
14.4.1 Drawing Controls with the Component Toolbox 14-10
14.4.2 Drawing Controls with Drag-and-Drop 14-11
14.4.3 Creating Autoload Controls 14-14
14.4.3.1 Creating an unpaged autoload control 14-15
14.4.3.2 Creating a paged autoload control 14-18
14.4.3.3 Modifying autoload controls 14-19
14.4.3.4 Understanding the generated code (unpaged controls) 14-20
14.4.3.5 Understanding the generated code (paged controls) 14-21
14.5 Configuring Control Properties 14-25
14.5.1 Associating Data with a Control 14-26
14.5.2 Layout Data Control Property 14-28
14.5.3 Controls: Related References 14-29
14.6 Positioning and Aligning Controls 14-30
14.7 Refining Your Screen 14-32
14.7.1 Basic Editing Commands 14-33
14.7.2 Determining Control Tab Order 14-34
14.8 Associating Code with Screen Elements 14-35
14.8.1 Entering the Event Editor 14-36
14.8.2 Event Procedures, Embedded Procedures, and Code Insertion Points 14-37
14.9 Working with Screen Templates 14-39
14.10 Generating a Screen 14-40
14.11 Testing Screens 14-41
14.12 Creating Portable Screens with AcuXUI 14-42
14.13 Creating a Logo Screen 14-43
14.14 Creating Dual User Interfaces 14-44

Chapter 15: Controls, Menus, and Toolbars

15.1 Introduction 15-2
15.1.1 Properties of Screen Elements 15-2
15.1.2 Events Overview 15-3
15.2 Standard Controls 15-3
15.2.1 Selector 15-4
15.2.2 Bar 15-5
15.2.3 Bitmap 15-5
15.2.4 Check Box 15-6
15.2.5 Combo Box 15-6
15.2.6 Date Entry 15-7
15.2.7 Entry Field 15-8
15.2.8 Frame 15-8
15.2.9 Grid 15-9
15.2.10 Label 15-12
15.2.11 List Box 15-12
15.2.12 Push Button 15-13
15.2.13 Radio Button 15-14
15.2.14 Scroll Bar 15-14
15.2.15 Status Bar 15-15
15.2.16 Tab 15-15
15.2.17 Tree View 15-17
15.2.18 Web Browser 15-20
15.3 ActiveX Controls 15-20
15.3.1 Adding ActiveX Controls to the Component Toolbox 15-21
15.3.2 Using ActiveX Controls in Your Screen 15-22
15.3.3 Removing an ActiveX Control from the Component Toolbox 15-23
15.4 Using the Menu Designer 15-23
15.4.1 Creating a Menu 15-24
15.4.2 Building the Menu 15-24
15.4.3 Adding Menu Functionality 15-26
15.4.4 Enabling the Menu 15-27
15.5 Using the Toolbar Designer 15-29
15.5.1 Creating a Toolbar 15-29
15.5.2 Using More Than One Toolbar 15-30

Chapter 16: Configuring the Report Composer

16.1 Introduction........16-2
16.2 Customizing the Report Composer Interface........16-2
16.3 Establishing Report and Control Defaults........16-4
16.4 Adding Report Templates........16-7
16.5 Report Keyboard Shortcuts........16-8

Chapter 17: Working with Reports

17.1 Introduction........17-2
17.1.1 Report Concepts: The Big Picture........17-2
17.1.2 Report Types........17-3
17.2 Adding a Report........17-4
17.3 Formatting the Report........17-5
17.3.1 Adding and Removing Report Sections........17-5
17.3.2 Configuring Report and Section Properties........17-9
17.4 Adding Report Controls........17-13
17.4.1 Using the Report Component Toolbox........17-14
17.4.2 Using Drag-and-Drop........17-14
17.4.3 Positioning and Alignment........17-15
17.5 Configuring Control Properties........17-16
17.6 Setting Print Conditions........17-17
17.7 Using Events to Populate a Report with Data........17-18
17.7.1 Adding Report and Report Element Events........17-19
17.7.2 Before and After Event Paragraphs........17-20
17.7.3 BeforeDoPrint and AfterDoPrint Event Paragraphs........17-21
17.7.4 LoadGridInit and LoadGridNext Event Paragraphs........17-22
17.7.5 Generating Report Files and Code........17-23
17.7.5.1 Reports Generated using BeforeDoPrint and AfterDoPrint events 17-24
17.7.5.2 Reports Generated without BeforeDoPrint and AfterDoPrint events........17-24
17.7.5.3 Grids Generated With LoadGridInit and LoadGridNext........17-25
17.7.5.4 Grids Generated Without LoadGridInit and LoadGridNext Events17-26
17.7.5.5 Existing Reports Imported from Earlier Versions of AcuBench....17-26
17.7.6 Working with data from multiple sources........17-27
17.7.7 Printing the Report........17-29
17.8 Creating Report Template Files........17-30
17.9 Sample Reports........17-31

17.10 Deploying in a Thin Client Environment....17-32
17.10.1 Using a Web Browser Control to Display and Print Reports....17-33
17.10.2 Using AcuBenchPrint.dll to Display and Print Reports....17-34

Chapter 18: The Report Controls and Property Reference

18.1 Introduction....18-2
18.2 Common Report Control Properties....18-2
18.3 The Report Box....18-6
18.4 The Report Check Box....18-7
18.5 The Report Date Time....18-8
18.6 The Report Entry Field....18-10
18.7 The Report Grid....18-12
18.8 The Report Image....18-14
18.9 The Report Label....18-16
18.10 The Report Line....18-16
18.11 The Report Occurs....18-17
18.12 The Report Radio Button....18-19
18.13 The Report Table....18-20

Chapter 19: Working with ACUCOBOL-GT Utilities

19.1 ACUCOBOL-GT Utilities....19-2
19.2 The Object File Utility....19-3
19.2.1 Object Libraries....19-3
19.2.2 Creating an Object Library....19-4
19.2.3 Retrieving Information About Objects....19-6
19.2.4 Generating Native Code....19-7
19.3 Using vio....19-9
19.3.1 Output Mode....19-10
19.3.2 Input Mode....19-12
19.4 Using logutil....19-13
19.5 Using vutil....19-15
19.5.1 Increasing the Maximum File Record Size....19-16
19.5.2 Examining File Information....19-17
19.5.3 Testing File Integrity....19-17
19.5.4 Rebuilding Files....19-17
19.5.5 Resetting User Counts....19-17
19.5.6 Creating Empty Files....19-18

19.5.7 File Size Summary....................19-19
19.5.8 Extracting Records....................19-19
19.5.9 Unloading to Other File Types....................19-19
19.5.10 Loading a File....................19-19
19.5.11 Converting Indexed Files....................19-20
19.5.12 B-Tree Listing....................19-21
19.6 Using acu4glfd....................19-21
19.7 XML Support in AcuBench....................19-22

Chapter 20: The AcuBench Integrated Debugger

20.1 The Debugger Interface....................20-2
20.1.1 Debug Menu and Toolbar....................20-3
20.1.2 Debugger Output....................20-4
20.1.3 Watch Window....................20-4
20.1.4 Stack Info Window....................20-6
20.1.5 Memory Window....................20-7
20.2 Debug Mode Compile Options....................20-7
20.3 Entering the Debugger....................20-8
20.4 Debug Menu Commands....................20-9
20.4.1 Tracing Functions....................20-10
20.4.2 Starting, Stopping, and Navigating the Debugger....................20-11
20.4.3 Debugger Scripts....................20-14
20.4.4 Breakpoints....................20-14
20.4.4.1 Setting a breakpoint using the Breakpoints dialog box....................20-16
20.4.4.2 Modifying a breakpoint using the Breakpoints dialog box....................20-17
20.4.4.3 Deleting breakpoints using the Breakpoints dialog box....................20-17
20.4.4.4 Viewing a breakpoint with the Breakpoints dialog box....................20-17
20.4.4.5 Monitoring variables using the Breakpoints dialog box....................20-18
20.4.5 Debugging Threaded Applications....................20-19
20.4.6 Quick Watch....................20-20

Chapter 21: Looking for Something?: Search and Replace

21.1 Introduction....................21-2
21.2 Find....................21-2
21.2.1 Locate a single instance of the search string....................21-3
21.2.2 Mark all instances of the search string....................21-3
21.2.3 List all occurrences of the search string....................21-4

21.3 Find in Files 21-4
21.3.1 Special Operators and Regular Expressions 21-5
21.3.2 Performing the Search 21-7
21.4 Find in Objects 21-8
21.5 Replace 21-9
21.6 Replace in Files 21-10

Chapter 22: Toolbar Reference

22.1 The Standard Toolbar 22-2
22.2 The Project Toolbar 22-4
22.3 The Editor Toolbar 22-6
22.4 The Debug Toolbar 22-6
22.5 The Align Toolbar 22-8
22.6 The Launch Toolbar 22-8

Chapter 23: Keyboard Shortcut Reference

23.1 Introduction 23-2
23.2 Main: Default Keyboard Shortcuts 23-3
23.3 Code Editor: Default Keyboard Shortcuts 23-24
23.4 Screen Designer: Default Keyboard Shortcuts 23-30

Appendix 24: Bringing Existing Code Into AcuBench

24.1 Introduction 24-2
24.2 Importing Data Definitions 24-3
24.3 Creating a PSF for an Existing Program 24-4
24.3.1 Prepare to Create the PSF 24-4
24.3.2 Create the PSF 24-5
24.3.3 Key-Status 24-6
24.3.4 Import Program Elements into the PSF 24-7
24.4 Adding Existing Code to an AcuBench Program 24-8
24.5 Updating a Character-based Screen 24-11
24.5.1 Benefits and Restrictions 24-12
24.5.2 Using the Character-to-GUI Wizard 24-14
24.5.3 Running under Windows 24-15
24.5.4 Setting Compiler Options 24-15
24.5.5 Setting the “--char2gui” Runtime Option 24-16

24.5.6 Screen Conversion....................24-17
24.5.7 Editing Screen Properties....................24-18
24.5.8 Manipulating the Screen in the Screen Designer....................24-21
24.5.9 Generating New Screen Section Code....................24-23
24.5.10 Integrating Code Back into Your Program....................24-23
24.5.11 Tips and Techniques....................24-28
24.6 Importing a Graphical Screen....................24-30
24.6.1 Setting Compiler Options....................24-31
24.6.2 Setting the "-import" Runtime Option....................24-32
24.6.3 Graphical Screen Importing....................24-33
24.6.4 Graphical Screen Importing Notes and Restrictions....................24-34

Index

1 Introduction

Key Topics

Product Overview 1-2
Companion Products 1-4
Technical Services 1-6

1.1 Product Overview

Welcome to the AcuBench® integrated development environment for COBOL from Micro Focus. AcuBench is a member of the ***extend***® family of interoperability solutions. AcuBench extends and enhances the ACUCOBOL-GT® development system with a powerful suite of graphical tools for data, program, interface, and report design. With AcuBench you can develop and maintain your COBOL applications in a developer friendly Microsoft Windows environment and deploy your applications on any of the more than 600 platforms supported by Micro Focus.

Some of AcuBench's major features include:

- a single, integrated user interface
- a powerful and flexible workspace, project, program development model
- extensive automated code generation capabilities that simplify the development of screens, reports, Working-Storage and Linkage section items, FDs and SELECTs, and other key program areas
- specialized designers and editors for creating and maintaining:
    - data layout definitions (file descriptors)
    - data set associations
    - event procedures
    - Working-Storage items
    - Linkage section items
    - ACUCOBOL-GT configuration files
- support for the ACUCOBOL-GT Thin Client technology, which lets you display your server-based application on graphical display hosts
- an integrated source level debugger
- graphical interfaces to ACUCOBOL-GT utilities
- support for many third-party version control systems
- a graphical interface that lets you design HTML and plain-text reports

AcuBench includes key project management supports for:

- concurrent development of multiple projects
- multiple project configuration management
- multiple project and file level compilation settings (modes)
- project level runtime and environment settings
- automated inclusion of COBOL COPY files
- access to external programs and utilities

Key code development features include:

- user configurable, COBOL-sensitive source code editor
- automated code completion capabilities (prompts for completing some phrases and easy insertion of template code)
- paragraph, variable, COPY file, and constant lists
- quick access to COPY files
- ability to bookmark code for ease of navigation
- search functions to find strings either in a specified directory structure or within a set of AcuBench objects

Key screen development capabilities include:

- support for ActiveX controls
- graphical and character-based screen designers
- templates for graphical and character-based screens, plus the ability to create additional user-defined templates
- graphical and character screen import capabilities
- customizable control defaults and Property windows
- in-line editing of event and embedded procedures

Integrated user interface

The workbench presents all of its primary capabilities in an integrated main application window. This user-configurable window contains several interior windows for viewing projects; working with data; working with programs and their components, including screens and reports; and for monitoring several types of system output (compilation messages, version control actions, search results, and more). There are also special windows for coding embedded procedures and working with the integrated debugger. You can control the visibility, size, and placement of all interior windows and toolbars.

Project-centered application development

A project model is used to support AcuBench application development. Each element of a COBOL application, large or small, is identified, added to, and tracked by AcuBench as a project. As a result, applications enjoy a standard structure that makes them easier to manage and maintain. This architecture allows AcuBench to handle many aspects of application development for you. For example, when you direct AcuBench to build (compile) a project, AcuBench automatically determines which elements of the project must be recompiled to make the project current.

Note: Unless otherwise indicated, the references to “Windows” in this manual denote the following versions of the Windows operating systems: Windows XP, Windows Vista, Windows 7, Windows 2003, Windows 2007, Windows 2008 R2. In those instances where it is necessary to make a distinction among the individual versions of those operating systems, we refer to them by their specific version numbers (“WindowsXP,” “Windows Vista,” etc.).

1.2 Companion Products

AcuBench is interlinked with other members of the ***extend*** family of solutions. The following sections briefly describe the technologies most closely related to AcuBench.

ACUCOBOL-GT

ACUCOBOL-GT is an ANSI 1985 COBOL compiler designed to provide a powerful development environment for a wide range of computers. Fast compile speed, clear error messages, and a multi-window source level debugger work together to provide a high performance, easy to use COBOL development platform. Portable object code, a generic interface to a variety of file systems, and a device-independent terminal interface help to simplify the distribution of applications developed with ACUCOBOL-GT.

In addition to portable object code, ACUCOBOL-GT can generate and execute object files that contain native instructions for specific types of processors. This enables you to optimize the use of CPU resources on the host machine while maintaining full portability within the same family of processors.

Complete access to the ACUCOBOL-GT development system, including all of its utilities, is available from within the AcuBench integrated development environment.

AcuConnect®

AcuConnect is a client/server technology that is an integral part of ***extend's*** distributed computing solution. AcuConnect lets you implement a client/server system in which the client piece can be as "thin" or as "thick" as you need.

Developers using AcuBench can take advantage of Micro Focus's Thin Client technology, allowing them to work in a user-friendly Windows environment and compile to and run from a UNIX, Linux, or Windows server. The thin client implementation of AcuConnect lets you run the user interface (UI) portion of your application on a graphical display host while the rest of the application and data reside on the server.

1.3 Technical Services

If you have a question about AcuBench, or if you encounter unexpected behavior during its use, ***extend's*** Technical Services specialists are ready to assist you. Before contacting Technical Services, please be prepared to provide the following information:

- the version of the Windows operating environment that you are using
- the version of AcuBench that you are using
- the version of the ACUCOBOL-GT compiler and runtime that you are using

You can display the AcuBench and ACUCOBOL-GT version information in AcuBench. From the Help menu select **About AcuBench**. In the information box, click the "Runtime version" and "Compiler version" buttons to display runtime and compiler version information.

For the latest information on contacting customer care support services go to:

http://www.microfocus.com/about/contact

For worldwide technical support information, please visit:

http://supportline.microfocus.com/xmlloader.asp?type=home

2 Getting Started

Key Topics

Introduction 2-2
About AcuBench Documentation 2-2
Related Documents 2-4
Notation 2-5
Sample Programs 2-5
System Requirements 2-6
Installing and Uninstalling AcuBench 2-6
Navigating the User Interface 2-7
Printing in the Workbench 2-12

2.1 Introduction

The following sections include several topics that can help you get started using AcuBench. A general description of the contents of the *AcuBench User's Guide* and some related documentation is provided, along with tips on the notation used in these documents. System requirements and installation procedures are defined, and the locations of some helpful sample programs are also given.

2.2 About AcuBench Documentation

All AcuBench functions and capabilities and directions for their use are documented in this *AcuBench User's Guide*. The topics in this guide are organized into the following general areas:

Introducing AcuBench	Chapters 1 through 5
Working with Projects	Chapters 6 and 7
Working with Data Files	Chapter 8
Working with Programs	Chapter 9
Program File Handling	Chapter 10
Editing Source Code	Chapters 11 and 12
Working with Screens	Chapters 13 through 15
Working with Reports	Chapters 16 through 18
Helpful Tools	Chapters 19 through 21
Reference	Chapters 22 and 23
Bringing Existing Code into AcuBench	Appendix A

The content of the on-line and print versions of this guide is virtually the same. In some cases, screen captures have been added to the printed book.

Accessing on-line documentation

You can access the on-line version of the *AcuBench User's Guide* by selecting **Help/AcuBench Manual** from the drop-down menu. The Help menu also provides shortcuts to the main page for the ***extend*8** documentation set, as well as other individual manuals and the ACUCOBOL-GT four-book manual set.

AcuBench also provides extended support for context-sensitive help. Click on or highlight the text, control, property, or other item about which you wish to learn more, then press **F1** in any part of the IDE. AcuBench opens the most appropriate page of the associated reference or user's guide that it can locate.

For example, if you are designing a screen and click on a push button control, then press **F1**, the ACUCOBOL-GT documentation opens in your default Web browser, showing the page from the *ACUCOBOL-GT User Interface Programming* manual that contains information about the attributes and properties of push button controls.

This context-sensitive help provides especially detailed information in the following interfaces:

- Code Editor
- Event Editor
- Screen Designer Property Window
- Project/Settings dialog

Note: Windows XP users who use Internet Explorer as their default browser and have Service Pack 2 installed should be aware that the operating system's default firewall protection may block operation of our online Help files. When a security alert appears, select **Unblock** or **Allow Blocked Content** to permit Help file operation.

If you are using Microsoft Internet Explorer, you can permanently disable the warning message on the Advanced tab of the Tools/Internet Options interface. Scroll to the "Security" options, then select **Allow active content to run in files on My Computer**.

Any time that a manual is open in your Web browser, you can use the full-text search facility to navigate to the information you need. If you have reached the manual through the context-sensitive help facility, you can bring up the table of contents by clicking the **Show In Contents** button at the top, right-hand corner of the documentation frame. To access information in other books in the ***extend*8** documentation set, click the **Books** button at the top, right hand corner of the page.

2.3 Related Documents

ACUCOBOL-GT and COBOL are documented in four manuals that together are called the ACUCOBOL-GT documentation set. The four-volume set is shipped on the product CD-ROM and also available under the support section of www.microfocus.com. The books in this set include:

Book 1, *ACUCOBOL-GT User's Guide*

This guide describes how to compile and run programs with ACUCOBOL-GT. Some chapters offer programming suggestions for both new and experienced programmers. The *ACUCOBOL-GT User's Guide* also describes the ACUCOBOL-GT runtime debugger and file utilities.

Book 2, *ACUCOBOL-GT User Interface Programming*

This book details ACUCOBOL-GT controls and discusses user interface development issues.

Book 3, *ACUCOBOL-GT Reference Manual*

This manual describes COBOL program structure and details every verb included in the language.

Book 4, *ACUCOBOL-GT Appendices*

This book includes host-specific information and descriptions of all library routines and configuration variables.

If you are using the print documentation, a fifth book, the *ACUCOBOL-GT Combined Indices*, contains a combined index for the entire four-book documentation set.

2.4 Notation

In this *AcuBench User's Guide*, menu selections are frequently shown as a series of strings separated by slashes. For example, you might see:

- To create a new file select **File/New**.

In the case of cascading menus, you would see:

- To turn on debugger file tracing, select **Debug/Trace Options/Trace File**.

Program names are shown in bold, such as **vutil.**

When citing a label or option that appears in a dialog box, the text is surrounded by double-quotes (""). For example: "Name of object file (-o)" is an option on the Project/Settings/Compiler dialog.

2.5 Sample Programs

The AcuBench product materials include two sample workspaces called "samples" and "reports". The "samples" workspace contains sample programs, COPY files, screens, resource files, and other program elements intended to demonstrate many of AcuBench's and ACUCOBOL-GT's capabilities. The "reports" workspace contains a number of graphical and text-only reports created using the AcuBench Report Composer. The sample workspaces are located in subdirectories of the installation directory under "Acucbl8xx\AcuGT\sample\acubench". We recommend that you familiarize yourself with the sample projects, programs, source code, and related files.

Other helpful sample programs are available for download in the Support area of the Micro Focus Web site.

2.6 System Requirements

To install and run AcuBench on your Microsoft Windows platform, your system must meet the following minimum requirements:

Hardware

- Intel Pentium III CPU, 300 megahertz; Intel Pentium IV, 2 gigahertz recommended
- 64 megabytes of RAM; 128 megabytes recommended
- 40 megabytes of available hard disk space; 120 megabytes recommended
- mouse
- 800 x 600 VGA display or better; 1024 x 768 DGA display recommended

Software

- Windows XP Professional Edition; Windows NT 4.0; Windows 2000 or 2003
- ACUCOBOL-GT Version 8.0 or later (compiler and runtime)

2.7 Installing and Uninstalling AcuBench

Before you install AcuBench, be sure that the host machine meets the minimum requirements specified in the previous section. Note that AcuBench Version 8.*x* requires the ACUCOBOL-GT Version 8.*x* compiler and runtime.

AcuBench is delivered on CD-ROM. In addition to the CD-ROM, you must have two license codes (the product code and product key) to successfully install the product. You should find the license codes in your product materials. If they are not included, please contact your Micro Focus ***extend*** Sales Professional.

The setup program starts automatically when you insert the CD-ROM into your drive. If the setup program does not start, go to the Windows Start menu, select **Run**, and enter the device name of your CD-ROM followed by "setup.exe". For example:

```
    D:\setup.exe
```

Follow the instructions on the screen and enter the license codes when prompted.

Caution: To avoid the unintentional overwriting and loss of existing AcuBench projects, files, and executables, install this version into a new folder.

AcuBench can be installed on systems that also have an earlier version of the workbench installed. You do not need to move or uninstall the earlier version in order to use that version and the new version. Note that the new version does not automatically find and use the ".ini" file options used by the earlier version. If you want the current version of AcuBench to use the options specified in a ".ini" file from a previous version, use the Tools/Options Load command to load the previous version's ".ini" file into the current version's Application Data folder.

If you reinstall the same version of AcuBench for any reason, note that the existing ".ini" file in the Application Data folder is not overwritten at installation. For example, if you want to revert to the installation defaults in the "AcuBench8xx.ini" file, you would need to delete the existing ".ini" file before reinstalling AcuBench.

To uninstall AcuBench, use the Add/Remove Program applet in the Windows Control Panel.

2.8 Navigating the User Interface

Access to most AcuBench functions is provided directly in the main application window. The size and layout of most screen elements is customizable. User interface elements, such as toolbars and sub-windows, can be moved, docked, undocked, made visible, or hidden at your discretion.

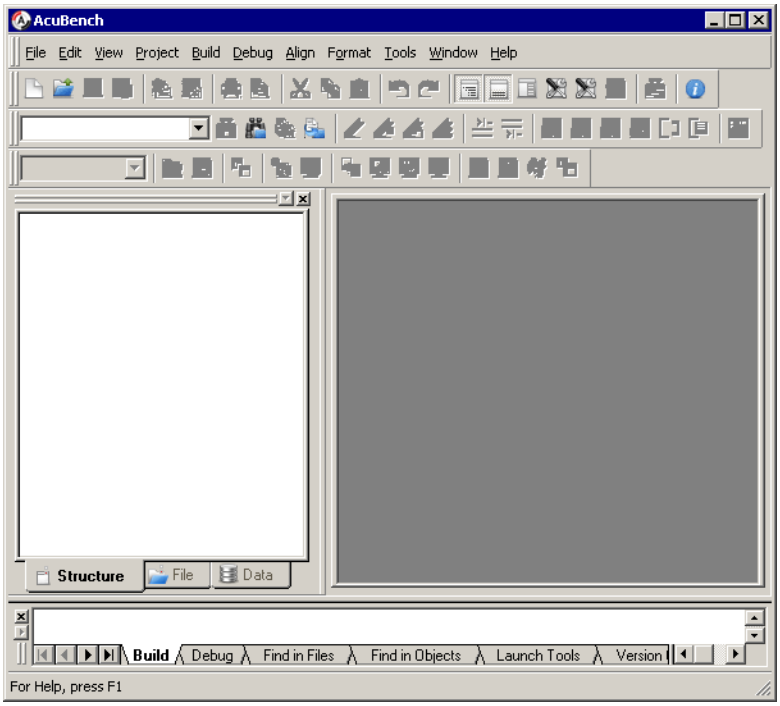
Position and visibility attributes are retained from session to session so that the main application window has the same configuration and appearance as when you ended your last session.

The **title bar** displays the name of the application (AcuBench), followed by the name of the workspace, the name of the current file, and the name of the current project. It also indicates whether your current project is using thin client functions.

The drop-down menus on the **menu bar** provide access to all workbench functions, configuration options, and on-line documentation. Access to ACUCOBOL-GT utilities, such as **AXDEFGEN**, **cblutil**, and **vutil**, is available on the Tools pull-down menu.

The AcuBench **toolbars** provide push button access to the most frequently used workbench functions. You can display or hide each toolbar via the View drop-down menu, or by right-clicking on a toolbar. You determine which toolbars display and which buttons appear on each toolbar using the Tools/Customize dialog, discussed in **Chapter 4, section 4.9, "The Customize Dialog."** This interface also allows you to create new, custom toolbars.

By default, each of these features appears at the top of the AcuBench interface when you open the application. The result is an interface that looks much like the following:.



2.8.1 The Toolbars

By default, AcuBench displays the Standard, Project, Editor, and Align toolbars at startup. When you use the AcuBench integrated debugger, the Debug toolbar is displayed, as well.

You can display, hide, resize, and move the toolbars. Toolbars that are not *docked* under the menu bar or to the edge of the window are called *floating* toolbars.

When you place the mouse pointer over a toolbar button for more than a second, a *tool tip* is displayed, showing the button's descriptive title. At the same time, a short description of the button's function is also displayed in the status bar.

Manipulating toolbars

To move a toolbar to a new position:

1. Place the mouse pointer within the toolbar, but not over a button.
2. Click with the left mouse button and drag the toolbar to the desired location.

To display or hide a toolbar, you can do any of the following:

- Click the toolbar's name in the View/Toolbars menu. If a toolbar is visible, a marked check box appears next to its name in the menu listing.
- Right-click on any toolbar, then select the name of the toolbar that you want to hide or display.
- Open the Tools/Customize interface and click the toolbar's name. If a toolbar is visible, a marked check box appears next to its name on the Toolbars tab.

To reshape a toolbar:

1. Position the pointer on the edge of the floating toolbar so that the pointer changes to a double-headed arrow.
2. Down click with the left mouse button and drag to alter the shape.

2.8.2 The Workspace Window

The Workspace window provides three different views of project components: the Structure view, the File view, and the Data view. Each of these views is represented by a tab at the bottom of the window; click a tab to switch from view to view.

These three views make up the primary interface for working with project elements in AcuBench, as follows:

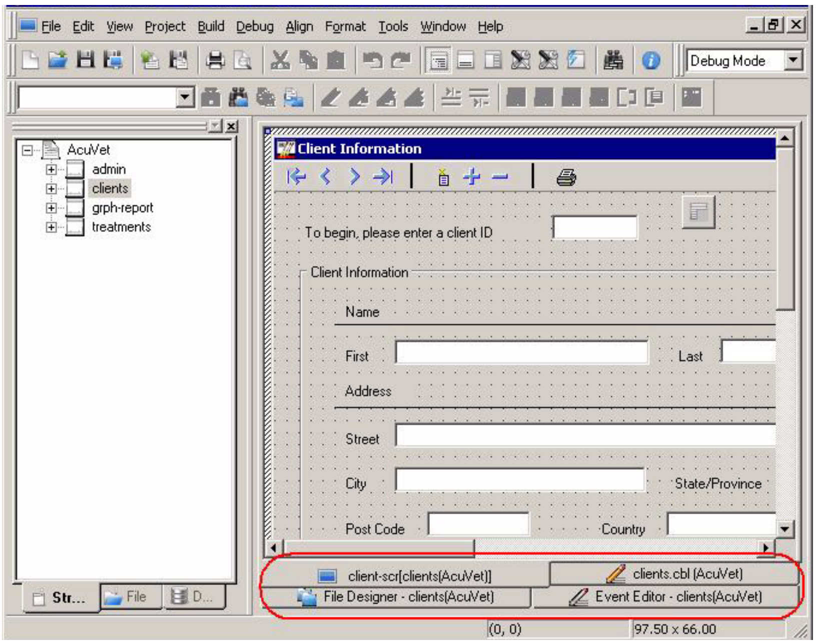
- The **Structure view** shows AcuBench *program structure files*, with interfaces to the graphical designers used to generate program code. Program structure files and the Structure view are discussed in **Chapter 9, "Chapter 9: Working with Programs."**

- The **File view** shows all physical files associated with an AcuBench project. Interacting with files and folders in the File view is discussed in **Chapter 6**, **section 6.3.3** through **section 6.3.6**.

- The **Data view** lists AcuBench *data layout files*, used to generate FD and SELECT statements and define file handling code at the project level. Data layout files and the Data view are discussed in **Chapter 8, "Chapter 8: Working with Data at the Project Level."**

2.8.3 The Development Window

All open project files, screens, and active editors (such as the Event Editor or File Designer) are displayed in the Development window. When you have more than one file or editor open in the Development window, each one is assigned to a tab, displayed at the top or bottom of the window.



To shift the focus from one file or editor to another in the Development window, click the appropriate tab. You can also use the Ctrl+F6 keyboard shortcut to scroll through open editors, or use the Window drop-down menu.

2.8.4 The Screen and Report Component Toolboxes

The AcuBench interface includes two toolboxes, one for the Screen Designer and one for the Report Composer. The Screen Component Toolbox contains the palette of ACUCOBOL-GT screen controls. If you have defined ActiveX controls for use with your project, these also appear in the toolbox. The Screen Component Toolbox is discussed in detail in **Chapter 14, section 14.4.1, “Drawing Controls with the Component Toolbox.”**

The Report Component Toolbox contains the “controls,” or elements, that you can use to design different sections of a report. Some report controls are similar to screen controls. The Report Component Toolbox is covered in **Chapter 17, section 17.4, “Adding Report Controls.”**

2.8.5 The Output Window

All workbench standard output is displayed in the Output window, which has five tabbed panes: Build, Debug, Find in Files, Launch Tools, and Version Control. Each pane captures the output messages related to its label. For example, the Build pane displays messages generated by parsing, building, and compiling. The Debug pane captures output generated from the debugger (but not program output).

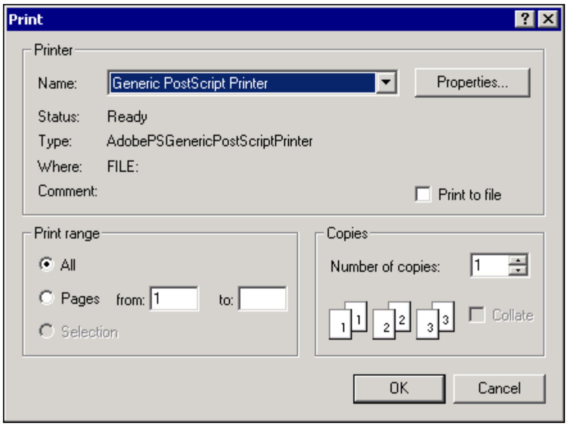
To print the contents of the current tab of the Output window, right-click inside the pane and select **Print**.

2.8.6 The Status Bar

The status bar displays various information pertaining to the current state and activities of the workbench.

2.9 Printing in the Workbench

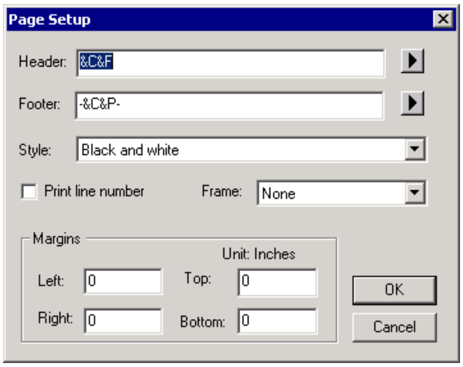
In addition to source code files and other text documents, AcuBench's print function allows you to print hard copies of screen forms, report forms, and the contents of the Output window. Select **File/Print** or click the Print icon on the Standard toolbar. This opens a standard Windows print dialog that allows you to select a printer, specify a page range, and so on.



Name: Generic PostScript Printer
Status: Ready
Type: AdobePSGenericPostScriptPrinter
Where: FILE:
Comment:
☐ Print to file
Print range: [✓ All] Pages from: 1 to: Selection
Number of copies: 1
☐ Collate

Use the AcuBench **File/Print Setup** command to change printers, select an output tray, change paper orientation, and so on. Like the Print dialog, the Print Setup dialog is a standard Windows interface.

When you print a text file, rather than a screen or report form, you can use the **File/Page Setup** command to customize the appearance of the printed page, including the header and footer, if any. This command opens a Page Setup dialog that also allows you to specify a page style, margin sizes, and other options.



Header: &C&F
Footer: -&C&P-
Style: Black and white
Print line number: ☐
Frame: None
Left: 0
Top: 0
Right: 0
Bottom: 0

The Header and Footer fields of the of the Page Setup dialog can take a text string or any of the special options described in the following table. You can either manually enter the code for any of the special options, or click the arrow button to the right of the field and select an option from the pop-up list.

Option	**Description**
Full Path Filename &U	Prints the full path and file name in the header or footer
Filename &F	Prints the file name in the header or footer
Page Number &P	Prints the page number in the header or footer
Current Time &T	Prints the time the document was printed in the header or footer
Current Date &D	Prints the date the document was printed in the header or footer
Left Align &L	Prints the header in a left alignment or footer
Center &C	Prints the header in a center alignment or footer

Option	**Description**
Right Align &R	Prints the header in a right alignment or footer

You can also set the following additional options to determine the appearance of your print output:

Option	**Description**
Style	Select any of the following styles from the drop-down list: **Black and White** prints your document in black and white. **Color** prints your document in color. **Zebra** prints your document using alternating gray and white stripes.
Print line number	Check this check box if you want the line numbers of your file to print on your document.
Frame	Select **None** if you do not want to print a frame around your document, or **Inner** to print a simple, single-line box around each page of text.
Margins	Allows you to specify margin settings for your printed document. Set the left, right, top, and bottom margins in measurements of inches.

3 Workbench Concepts

Key Topics

AcuBench Concepts 3-2
Project Management, Organization, and Structure 3-2
Automatic Code Generation 3-6
AcuBench File Types 3-10
Development Approaches 3-12
Using Thin Client Technology with AcuConnect 3-24

3.1 AcuBench Concepts

Several fundamental concepts shape AcuBench's organization, structure, and approach. These ideas include:

- an organizational model that groups programs within projects and projects within workspaces
- an emphasis on automated code generation

These concepts, as well as a discussion of development approaches, are described in the sections that follow.

3.2 Project Management, Organization, and Structure

In AcuBench, application development is organized into three levels of structure: workspace, project, and program.

The central unit is the *project*. An AcuBench project includes all of the files and resources needed to build, test, and deploy an application. Every project has its own compile, runtime, and environment settings (in fact, projects can have many sets of settings, called *modes*). Every project is autonomous, although it can coexist and share resources with other projects.

To provide greater development flexibility, projects inhabit a larger working environment called the *workspace*. The workspace hosts one or more projects and provides a common work area that supports the concurrent development of multiple projects. The workspace layer also makes it possible to standardize AcuBench default settings (for code generation, graphical user interface and report design, and general environment behavior, among other options) across projects.

A Workspace window displays components of each project in the workspace, providing easy and organized access to project components, options, and settings. Through this window, you can instantly switch between projects and quickly copy components from one project to another. Each project's state information is preserved and each project's unique settings (compiler, runtime, and environment options) are maintained. Projects can, but do not have to, share files. Projects are not formally related in any way, other than belonging to the same workspace.

Every project has at least one *program* and can have many programs. A program may be a single COBOL source file, a source file and its COPY files and resources, or a source file augmented by a *program structure file*. When you work in AcuBench, the program structure file is what makes it possible to use the many graphical design tools and editors (excluding the Code Editor) designed to streamline code development. Only programs that include a program structure file appear in the Workspace window's Structure tab.

We recommend that all AcuBench users read **section 3.5, "Development Approaches."** If you intend to use AcuBench's design tools and code generation facilities, it is also important that you read **section 3.3, "Automatic Code Generation."**

The workspace, project, and program concepts are discussed in more detail in the sections that follow.

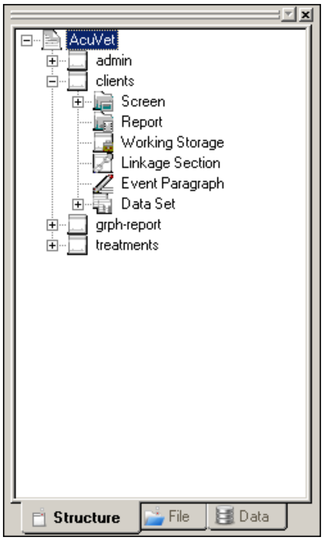
3.2.1 The Workspace

In AcuBench, each project and all of its associated files and resources belong to a workspace. Put another way, a workspace is a container for projects. The workspace is used to keep track of member projects and their files, options, and settings. It is also an interface that organizes, displays, and allows for interaction with project elements. The workspace is implemented in the form of a workspace file and a Workspace window.

The workspace file is created and maintained by AcuBench. In it is a list of member projects, project files, and project compile, runtime, and environment settings. Workspace files have the suffix ".pjt". You should never directly edit a workspace file.

The Workspace window is a sub-window of the AcuBench main application window. It presents each project's components in several *views*. You can interact with the project elements displayed in the Workspace window in many ways. Each node in the window has a context-sensitive right-click pop-up menu that offers a selection of actions appropriate for the associated object.

The Workspace window has three tabs at the bottom that allow you to select from among three workspace views. Each view uses a Windows Explorer-style *tree view* to display its contents.



The Workspace window

Only COBOL programs that have a program structure file appear in the workspace *Structure* view. The Structure view displays access points to program areas that have a corresponding workbench development tool, such as screens (Screen Designer), reports (Report Composer), Working-Storage (Working Storage Editor), Linkage section (Linkage Section Editor), Event paragraphs (Event Editor), and data sets (Data Set Designer).

If you are importing an existing program, you do not have to create a program structure file. However, if you do not, your program is not displayed in the Structure view, and you cannot use the workbench tools that generate code.

The workspace *File* view displays each project's member files. Project files are grouped into related types, such as Source, Screen, Report, Copy, Object, Remote Object, List, Resource, and FD files. Files can be opened or executed by double-clicking on their name or icon. You can also define additional file groupings (see **section 6.3.3**).

The workspace *Data* view shows each project's file descriptors (also called data layouts). File descriptors are created with the workbench File Designer. For information about the File Designer, see **Chapter 8, "Chapter 8: Working with Data at the Project Level."**

The workspace concept and Workspace window simplify many aspects of working with project components. By double-clicking on nodes in the various views, or by selecting options from the right-click pop-up menus, you can initiate most common workbench activities.

A *workspace information file* (".wif") is used to record workspace configuration and state information. The workspace information file is a binary file that is maintained by AcuBench in the same directory as the workspace project file (".pjt").

3.2.2 Projects

A project is the fundamental component that you create when you start to develop a program under AcuBench. Once created, you populate the project with all of the files and resources needed to build, debug, and deploy one or more programs.

You can create multiple projects in the same workspace. When a workspace is open, all of its member projects are displayed in the Workspace window.

You can have multiple programs in the same project. However, compiler, runtime, and environment settings are generally made at the project level. While it is possible to set compiler settings for each individual program in a project, two programs that have different runtime or environment settings should probably be placed in separate projects.

You can use the File/Save Project function to export information about an existing project into a project file (".pjf"). This file retains all project-related information that is stored in a workspace file and replaces all relative path names with full path names. A project saved as a ".pjf" can be physically separated from the workspace where it resides, facilitating the sharing of projects among development team members. Note, however, that once a ".pjf" is created, it is static. In other words, you have to manually select the Save Project option to update the contents of a ".pjf" file.

If you are going to transfer projects between workspaces, you can use the Tools/Options/Environment/General interface to set an option that directs the workbench to automatically copy project files to a new project directory when you add a project to a workspace. This option is also available in the Open Project dialog.

3.2.3 Programs

Because of the sophisticated code generation capabilities of AcuBench, the definition of what constitutes a program has grown. Without AcuBench, a program is, at its simplest, a COBOL source file and its COPY files (if any). In AcuBench, a program can still be just a COBOL source file and its COPY files, but it can also be something more.

If you plan to use any of AcuBench's automated code generation facilities, your program must have a program structure file. Programs created in AcuBench using the File/New/Program command are automatically give a program structure file. For information about using AcuBench's code generation with source code created outside the workbench, please see **Appendix 24**.

All of the development work that is done with workbench tools that generate code, such as the Screen Designer or File Designer, is stored in the program structure file. When you generate code for the program, based on the information in the program structure file, several COPY files are created and some code may be generated directly into the program file (you have control over what files are created and what code is generated). It is important to understand that vital program information is stored in the program structure file (in a non-COBOL representation), and that a program that uses automated code generation facilities cannot be constituted without it. In a very real sense, the program structure file is as much a part of the program as the COBOL program source file (".cbl").

For a discussion of program development approaches, see **section 3.5**.

3.3 Automatic Code Generation

The Workspace Structure view provides a visual interface to most of AcuBench's automatic code generation facilities. Each icon in the program tree that appears in the Structure view—Screen, Report, Working Storage, Linkage Section, Event Paragraph, Data Set—can be used to open a graphical designer, which stores its code in a program structure file. When you select a program in the Structure view and select the **Generate** command, AcuBench opens the program structure file and uses that file to create screen, report, Working-Storage, Linkage, event, and file-handling COBOL code.

The COBOL code that AcuBench generates from the program structure file is, by default, placed into a ".cbl" source file and several COPY files. Tags in the source file tell AcuBench where to place generated code. The area outside of these tags is reserved for manual editing. This means that, by default, all code between the AcuBench tags is deleted and recreated every time you generate code, but code that you have added outside of the tags is preserved. The various AcuBench tags and their purposes are discussed in **Chapter 4, section 4.6.2, "Program Tag Options."**

Changes made to a single AcuBench designer may cause code to be generated in several COPY files. When you use the Screen Designer to add a status bar control to a graphical screen, for example, AcuBench's default code generation produces a combination of Screen Section, Working-Storage, and Procedure Division code in ".scr", ".wks", and ".prd" COPY files. Ways to customize code generation behavior and affect the naming and use of COPY files are discussed in **section 3.3.2, "Controlling code generation,"** and **Chapter 4, section 4.6.1, "Generate Document Options."**

The program structure file also makes it possible for AcuBench to handle sometimes subtle interactions between the different graphical designers. When you create an entry field control in the Screen Designer, for example, AcuBench creates a Working-Storage variable to hold the contents of that field. The variable appears in the Screen Designer Property sheet for the control and in the Working-Storage Designer's graphical interface. When code is generated, Screen Section code associates the variable with the control's VALUE property and the variable definition appears in the Working-Storage Section.

If you want AcuBench to generate file-handling code, you need to create one or more *data layout files*, in addition to the program structure file. A data layout file is an AcuBench-specific file from which file descriptions, SELECT statements, custom extended file descriptors, and sort descriptions are created. For each data file that you plan to use with a project, you create one data layout file. Data layout files are discussed in more detail in **Chapter 8, section 8.2, "Defining Data Files for Use in Projects and Programs."**

Once you have created a data layout file at the project level, you can define *data sets* for each program in the project. A data set is a program-level definition specifying how an individual program will use a given data file. Based on your specifications, AcuBench uses the data set definition to determine what kind of file-handling code (OPEN, READ, and WRITE paragraphs, for example) to generate in the Procedure Division. Data sets are discussed in more detail in **Chapter 10, section 10.2, "Using the Data Set Designer."**

3.3.1 Generated COPY Files

Depending on the workbench tools used to develop the program, AcuBench may generate the following files:

1. *program*.scr (screen COPY file), containing:

    - Screen section description entries
    - If needed, CHARACTER or GRAPHICAL syntax to distinguish between graphical and character-based screens with the same name
    - Syntax to support event procedures, exception procedures, and/or embedded (before/after) procedures, when defined

2. *program*.mnu (menu COPY file), containing menu bar descriptions and calls to W$MENU.

3. *program*.evt (event and embedded procedure COPY file). This file contains code that you add in the Event Editor and code that AcuBench generates for event and exception handling. The portion of the code that is created by AcuBench does not appear in the Event Editor and is not intended for modification.

4. *program*.prd (Procedure Division COPY file), containing:

    - Program initialization and exit routines
    - If one or more screens has been created in the Screen Designer, screen-handling routines
    - If a screen uses exception or termination values, an exception-processing routine.

5. *program*.wrk (Working-Storage COPY file), containing:

    - Variables defined for control properties
    - Variables defined in the Working Storage Editor
    - When a data set is defined for the program, the file status data item

6. *program*.lks (Linkage Section COPY file), containing variables defined in the Linkage Editor

7. *program*.cbl, a source code file in which most code is generated between tag sets. This source file includes:

    - Attributes and comments from the Program Properties dialog
    - COPY statements for generated files

3.3.2 Controlling code generation

To refine your control over automated code generation, you can:

1. Set code generation options in the Tools/Options/Code Generator dialog.

2. Add or remove tag pairs from the program. The workbench cannot insert code into areas that are not tagged. If you remove a tag pair, AcuBench does not insert code in that area. Code generation is restored to an area if the tags are reinserted.

    You can control whether specific tag pairs are automatically generated or suppressed via the Tools/Options/Code Generator/Program Tag interface. For more information about this dialog, refer to **section 4.6.2, “Program Tag Options.”**

For information on the following facilities, see:

File Designer	Chapter 8
Data Set Designer, Working-Storage Editor, Linkage Editor	Chapter 10
Screen Designer	Chapters 13 and 14
Event Editor	Chapter 14

Note that to use these facilities, your program must have a program structure file. If you want AcuBench to generate COPY statements into the program file, you must upgrade your program to include AcuBench tags.

3.4 AcuBench File Types

A number of different file types are created and used by AcuBench. Some of these files are COBOL COPY files, produced by AcuBench's code generation facility, as discussed in the previous section. Others, however, are non-COBOL files used to define AcuBench workspaces and projects, store workspace state information, determine basic workspace settings, define template information, transfer compiler and runtime settings between projects, generate programs and data handling code, and so on.

Especially when you are using a version (source) control system, as discussed in **Chapter 5**, it can be vital to know what each file contains and how it is used by AcuBench. Because this information is particularly useful in a version control context, the reference includes two tables: one listing files generally tracked by version control, and one listing files that do not need to be tracked (either because the information they contain is relevant only to a single user, or because the files are completely recreated on generation from another file). Note that projects developed in the workbench may contain additional file types that are not AcuBench-specific (such as ACUCOBOL-GT ".def" or ".acu" files, user-defined COPY files, and resource files).

Commonly tracked files:

Extension	**Description**
.pjt	The workspace project file includes information about the projects within a workspace and the location of programs within each project.
.pjf	The individual project file contains the location of all programs within a single project. By default, project information is stored in and read from the workspace project file (".pjt").
.psf	The program structure file contains the information AcuBench needs to generate a program.
.cbl	Default extension for COBOL source code. Note that source generated by AcuBench is recreated from the ".psf" file when you generate.
.dlt	Data layout files contain the definitions that AcuBench uses to create FD, SL, and SD COPY files, as well as user-defined file handling code.
.ini	AcuBench ".ini" files contain information about default program and control properties and code templates, as well as information about the overall appearance of the workspace and its editors.
.pof	Project option files contain project settings, including compiler and runtime flags, AcuBench environment settings, and library options.
.stf	Screen template files can be used to create a consistent look and feel for a user interface.
.prf	Program option files store the default properties assigned to new AcuBench programs, including key status values and code generation options (shown in the Program Properties interface).
.wtf	Report template files can be used to create a consistent look and feel for reports across an application or set of programs.

Files that are not usually tracked:

Extension	**Description**
.wif	The workspace information file contains the current state of the AcuBench workspace (which windows are open, which breakpoints are set, and so on).

Extension	**Description**
.evt	Event paragraph COPY file, generated from the ".psf" file and accessed through the Event Editor.
.lks	Linkage section COPY file, generated from the ".psf" file and accessed through the Linkage Editor.
.mnu	Menu COPY file, generated from the ".psf" file and accessed through the Menu Designer dialog in the Screen Designer.
.prd	Procedure Division COPY file, generated from the ".psf" file. The code that is generated into this file comes from a number of the AcuBench editors and designers, including the Screen Designer and File Designer.
.rpt	Report COPY file, generated from the ".psf" file and accessed through the Report Composer
.scr	Screen Section COPY file, generated from the ".psf" file and accessed through the Screen Designer.
.wrk	Working-Storage COPY file, generated from the ".psf" file and accessed through the Working-Storage Designer.
.fd	File description COPY file, generated from the ".dlt" file and accessed through the File Designer.
.sl	SELECT statement COPY file, generated from the ".dlt" file and accessed through the File Designer.
.sd	Sort COPY file, generated from the ".dlt" and accessed through the File Designer.

3.5 Development Approaches

AcuBench supports a wide variety of approaches to developing and maintaining COBOL applications. The approach that you use is determined by the set of workbench tools you employ, the needs of your application, and your approach to software development. If you plan to use workbench tools that automatically generate code, such as the Screen Designer and File Designer, some elements of your program's content and structure are determined by the methods used by those tools. For a description of the workbench's approach to automated code generation, see **section 3.3, "Automatic Code Generation."**

The following sections discuss general development approaches. Included in each discussion are specific instructions on how to establish a workbench project that best supports that development approach.

The first section focuses on applications that are new or migrating to AcuBench, and that intend to make use of the workbench's automated code generation tools.

The second section focuses on the "traditional" development approach, which foregoes the use of any of the workbench's automated code generation facilities (no use of the Screen Designer, File Designer, Data Set Designer, Working Storage Editor, Linkage Editor, or Event Editor). The developer must write every line of application code. This approach does not restrict the use of non-code generating workbench facilities in any way.

The third section provides additional tips for developers working with large applications, made up of a great many programs and COPY files. It focuses on maximizing performance while taking advantage of AcuBench's code generation tools.

3.5.1 General Considerations

Regardless of the approach you use, there are several details of project setup that you should attend to every time you create a new project. Each item below is discussed in detail in the sections that follow.

- You should check and adjust, if desired, the project prefix settings.
- If you change the default name of a project subdirectory, you may need to check and adjust certain project settings.
- Consider placing your project files directly in the project directories.
- You may want to customize the default keyboard shortcut settings.
- You should review all compile, runtime, and environment settings and, if possible, test your project setup by compiling and running the application before you begin making significant changes to your application code.

3.5.1.1 Project prefix settings

Before you create a new project, open the Options dialog (select **Tools/Options**) and examine the default prefix values in Environment/Prefix. The prefix values are the various strings used to determine the base name for new files created by the workbench. For example, the default program prefix is “Program.” When you create a new AcuBench program, the workbench takes that prefix and appends a number to create a unique program name. As a result, if you accept all of the default names suggested by AcuBench, your first program is called “Program1,” the second is called “Program2,” and so on. Note that we do recommend renaming your programs as you create them to assign more descriptive names.

The prefix values are workbench-level settings (as opposed to workspace or project level settings) and are applied to all new projects and files.

3.5.1.2 Project directories

When you create a project, you can change the name of any project subdirectory. You can choose to omit a subdirectory altogether (by making that entry field blank). To modify or eliminate a subdirectory for an individual project during project creation, select the **More Info** button on the File/New/Project dialog box.

You can also change the base directory structure used for all new AcuBench projects in the Tools/Options interface under Environment/Prefix. Changes to the “Working directories” area affect the names of the directories used to hold the different sorts of files associated with a project by default. For more information, see **Chapter 4, section 4.3.7, “Prefix Options.”**

Note that renaming or eliminating a subdirectory does not change the name of the corresponding folder in the Workspace window. However, the logical link between the Workspace folder and the corresponding directory is updated so that the correct related files are displayed.

If you change a name or eliminate a directory, you may need to adjust some of your project settings. To view and set these options, right-click the project node in the File view and select **Settings** from the pop-up menu. This opens the Project Settings dialog, discussed in detail in **Chapter 7**.

- If you change the name of the "object" directory, or eliminate it, you may need to change the argument for "Name of object file (-o)" in the Standard Options catalog of the Compiler tab.

  In most instances, this should be set to: **%objectdir%\@.acu**.

- If you change the name of the "list" directory, or eliminate it, you may need to change the argument for "Name of list file (-Lo)" in the Listing Options catalog of the Compiler tab.

  In most instances, this should be set to: **%listdir%\@.lst**.

Note: For more information about %objectdir%, %listdir%, and similar macros, see **Chapter 4, section 4.3.7, "Prefix Options."**

3.5.1.3 Placing files in the project directory

When you use the Add/Remove Files function to add files to your project, you can specify files that are located on your hard drive or on any network drive that your Windows operating system recognizes. AcuBench creates a logical link to the file so that it can be accessed through the File view of the Workspace window while remaining in its original location on disk. To facilitate use of a source control system, or to make it easier to move a project from machine to machine, you may instead want to copy or move files into your AcuBench project directory structure before adding them to the project.

3.5.1.4 Keyboard shortcut settings

The workbench offers a great deal of flexibility in the definition of keystroke shortcuts. You can even define the same keystroke combination to perform different tasks depending on whether you're working in the Screen Designer or the Code Editor. Keystroke shortcut definitions are defined for AcuBench as a whole, rather than for individual workspaces. It is possible, however, to create multiple settings files (specifically, ".ini" files) and then load the appropriate file according to the task you wish to perform or the workspace or project with which you plan to work (see **section 4.2**). For step-by-step instructions on how to change keystroke definitions, see **section 4.3.6**. For a complete list of workbench commands and default keystroke shortcut definitions, see **Chapter 23, "Chapter 23: Keyboard Shortcut Reference."**

3.5.1.5 Setting options and testing

Soon after the new project is created, it is a good idea to open the Project Settings dialog to review the settings of all compiler and runtime options. You should also review the definitions of the environment variables.

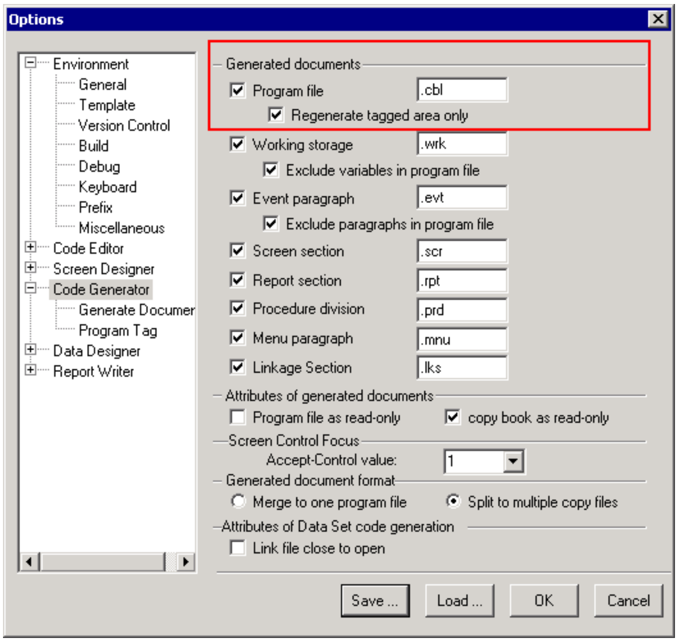
After you have populated the project with the application files, and after you have reviewed the compiler and runtime switches, but before you start making changes to your program, it is a good idea to attempt to compile and run the application or a sub-component of the application.

Caution: If you are going to use workbench tools that automatically generate code, there are some circumstances in which you should be careful to protect existing files from being overwritten. Please read the following information carefully to understand the risks.

When the workbench generates code, except when working with the ".cbl" file (discussed below), it completely rewrites the target file. All files that the workbench creates are given names based on the program name, as discussed in **section 3.3.1**. If you have a file in your project with the same name as one that might by generated by the workbench, you should rename the file.

You can determine which files are generated by changing the settings in the Tools/Options/Code Generator/Generate Documents interface.

The ".cbl" file is a special case. Whenever you have a program with a program structure file, AcuBench tags are added to the ".cbl" file for code generation purposes. By default, in the Tools/Options/Code Generator interface, under Generate Documents, the first and second check boxes on the page are marked. Marking the first check box, "Program file," determines that AcuBench will use the tags in the ".cbl" source file to generate code into the source code document. If you de-select this option, the currently existing ".cbl" file is not changed in any way going forward. The second check box, "Regenerate tagged area only," indicates that only the tagged areas of the ".cbl" file are changed. Any code that you have manually added to the ".cbl" source file is preserved during code generation as long as this box is marked.



Generated documents
[✓] Program file .cbl
[✓] Regenerate tagged area only
[✓] Working storage .wrk
[✓] Exclude variables in program file
[✓] Event paragraph .evt
[✓] Exclude paragraphs in program file
[✓] Screen section .scr
[✓] Report section .rpt
[✓] Procedure division .prd
[✓] Menu paragraph .mnu
[✓] Linkage Section .lks

Attributes of generated documents
[ ] Program file as read-only [✓] copy book as read-only

Screen Control Focus
Accept-Control value: 1

Generated document format
Merge to one program file [✓ Split to multiple copy files]

Attributes of Data Set code generation
[ ] Link file close to open

If you choose to generate the program file but **do not** choose the "Regenerate tagged area only" option, your ".cbl" file will be **completely overwritten** when you generate code. It is therefore very important that you pay careful attention to your Generate Document settings.

3.5.2 Developing New Applications

If you are developing a new application and intend to use AcuBench tools that generate code, such as the Screen Designer or File Designer, there are several issues that you should consider before you create your project. Of tantamount importance is that you read and become thoroughly acquainted with **section 3.3, "Automatic Code Generation."** Section 3.3 provides essential information about the AcuBench code generation model, and it outlines the associated program organization and structure.

When creating a new project:

1. Use the Tools/Options interface to set up code generation behavior, establish guidelines for building new project directory structures, and create the ".ini" file that will be used to standardize the development environment for the members of your development team. See **section 4.2, "The Tools/Options Dialog,"** for more information about the Tools/Options interface. For information about the importance of the ".ini" file, see **Chapter 5, "Chapter 5: Version Control."**

2. Use the File/New/Project command to create, name, and assign directories for a new project. Detailed, step-by-step instructions for creating a new project in either a brand new workspace or an existing workspace can be found in **section 6.3.1**.

3. Establish the runtime, compiler, and environment settings that will be used by programs in this project. Multiple sets of settings, called *modes*, can be created for each project. In addition, modes created in one project can be imported by other projects. For more information, see **Chapter 7, "Chapter 7: Project Settings."**

4. Determine which data files will be used by programs in the project and create data layout files to define those data files for use with AcuBench code generation. When you create data layout files, AcuBench code insight features as well as the Drag-and-Drop interface for quick screen and report design, among other features, are made significantly more useful. The process of creating data layout files is discussed in **Chapter 8, section 8.2, "Defining Data Files for Use in Projects and Programs."**

5. Create new programs for your project with the File/New/Program command. This command creates a program structure file. When you generate the program for the first time, a ".cbl" source file is created.

   You can also add existing program structure files, COBOL source files, COPY files, resources, and so on to your project. All of these options are discussed in **Chapter 6, "Chapter 6: Working with Projects,"** and **Chapter 9, "Chapter 9: Working with Programs."**

3.5.3 Developing Programs That Do Not Use Code Generating Tools

This is the approach that is most like traditional COBOL program development, except that you're working in a sophisticated integrated development environment. This approach foregoes the use of the Screen Designer and other AcuBench tools that generate code. Therefore, these projects do not need or make use of program structure files, nor do they make use of the Structure view of the Workspace window (project files are accessed in the File view). You, and other members of the development team, are responsible for writing all COBOL code. This approach in no way restricts your use of the ACUCOBOL-GT language, or the non-code generating workbench tools, such as the Code Editor's code completion facilities (see **section 12.5.4**) or the Configuration File Editor, to name only two.

There is nothing unusual about creating a project for the traditional development approach. For step-by-step instructions, see **section 6.3.1**. There are, however, a couple of important details to remember.

1. When you create the project, use the "Blank" project template. The "Blank" template does not create a program structure file (which you do not need).

2. When you use the Add/Remove Files interface to select the COBOL source files that you will be adding to your project, make sure that you have *not* marked the option to create a program structure file for each program. In other words, look at the "Files in project" frame and verify that the check box in the "Create .PSF" column for each program that you intend to add to your project is *not* marked.

   If you don't intend to create a program structure file for any of your existing programs, you can disable the "Create PSF" option at the workspace level. In the Tools/Options/Environment/Miscellaneous interface, locate the "Automatically create a program structure file" area and *deselect* the "When adding a source file" check box. When you do this, all "Create PSF" check boxes in the Add/Remove Files dialog are unmarked by default.

3.5.4 Working with Large Applications

Although there is no hard limit on the number of programs that can reside within an AcuBench workspace, when you are developing a large application, it is generally more efficient to use multiple small workspaces.

In AcuBench terms, a "large" application is defined by the combination of:

- The number of lines of generated code
- The number of controls on screens and in reports
- The number of Working-Storage variables maintained by AcuBench
- The number of external COPY files parsed at generation time

This means that if you are using AcuBench *without* code generation, you can have a great many programs without affecting the performance of the workbench. (Note that you may still want to divide your application between multiple workspaces for ease of maintenance.) But as you design graphical interfaces in AcuBench and make use of the workbench's code generation and automation features, it is increasingly important to keep your workspaces small and efficient.

Regardless of the composition of your large application, any time you have multiple developers working in AcuBench, it is important to use version (source) control software. See **Chapter 5, "Chapter 5: Version Control,"** for more information.

3.5.4.1 Factors that affect performance

When you expand a program in the Structure view of the AcuBench workspace, the program is loaded into memory. Programs are also loaded into memory at generation time (if they are not already resident in memory). This means that when you generate a workspace, every program in the workspace that has a ".psf" is loaded into memory. Once loaded, programs remain in memory until you exit AcuBench. This is done to speed up code generation and other automation features of the IDE.

Some of the automation features include the following:

- Automatic generation of Working-Storage variables for screens
- Automatic check for variable definitions within external COPY files before adding new variables to Working-Storage

When a program is loaded, AcuBench searches the program's screens and reports for properties that should be defined in Working-Storage. If AcuBench cannot find those definitions in any COPY file or FD, those properties are automatically added to Working-Storage. This feature:

- Makes it easy to recover when a variable is accidentally deleted
- Helps to avoid compiler errors
- Extends AcuBench's screen and report management capabilities

The feature is also CPU-intensive and memory-intensive. As a result, the feature can be disabled using the "Force variable check on load" option in the Tools/Options interface (see **Chapter 4, section 4.3.1, "General Environment Options"**). However, we recommend creating smaller projects and taking advantage of AcuBench's automation tools.

3.5.4.2 Organizing your workspace

As you work in AcuBench, remember that workspace organization is entirely virtual, and that an AcuBench project is a *virtual* grouping of related programs. This means that although AcuBench creates a default directory structure for a project, only AcuBench-generated code needs to reside within that structure. Remember that:

- Your project directory structure can be as simple as a single folder
- Multiple projects can share a single project directory
- Programs and COPY files can reside outside of the project folder

All of this makes it possible for you to manage your entire large application within a single *version control* project, while maintaining multiple *AcuBench* projects.

The AcuBench ".ini" file provides a means of standardizing screens, reports, code generation options, and other properties of the workbench environment across multiple workspaces. For more information, see **Chapter 4, section 4.2, "The Tools/Options Dialog."**

At the project level, project options files (".pof") allow you to create compiler, runtime, and environment settings once, then import them into other new or existing projects. See **Chapter 7, "Chapter 7: Project Settings,"** for more information.

3.5.4.3 The command-line interface to AcuBench

To make it easier to manage multiple small projects, AcuBench has a command-line interface that allows you to generate and build workspaces from the Windows **cmd** shell.

Note: The Windows "Command Prompt" shortcut launches the 16-bit **command.com** shell. The AcuBench command-line interface does not function in this environment. Instead, open the Start menu and select **Run**. In the Run dialog, type **cmd** and press Enter to launch the 32-bit shell.

The AcuBench command-line interface allows you to generate and build an entire workspace, or any program within a workspace. By default, whichever project mode is currently active for the project (as recorded in the ".pjt" file) is used to determine the compiler settings used for the build.

Tip: To ensure that you know which project mode you are using, you can create multiple copies of the PJT file. To do this, set a project mode (i.e., Release) in AcuBench, then save and close the workspace. Make a copy of the PJT file and give it a new name ("Prj1-Release.pjt"). Repeat the process for each of the project modes that you want to invoke from the command line. When you want to build a project in a particular mode, refer to the appropriate PJT file.

It is important to remember that only a single instance of each AcuBench version can run on a given machine. This means that you must close the IDE before using the command-line interface.

Building a workspace

The syntax for building a workspace using the command-line interface is as follows:

```
[acupath]\acubench80.exe /build [projectdir]\[workspace.pjt]
```

For example, if you have AcuBench installed in the default directory, and you would like to generate and build a workspace called "Project1", use the following command:

```
c:\Acucorp\Acucbl8xx\acubench\acubench80.exe /build c:\project1\project1.pjt
```

From the command-line interface, the "/build" and "/rebuild" flags are interchangeable. With either parameter, AcuBench checks whether code has changed since the last build. If any part of the program has changed, the code is regenerated and compiled. This is the same behavior that you get when you execute a Build Workspace command from the AcuBench Build menu.

The log file

All output messages generated during the build are recorded in a file called "build.log", placed in the project directory for the workspace being built. If this log file does not exist, it is created. If the log file already exists, messages generated by the new build are appended to the existing log. The log file also includes the start and end times for the build.

Building a single program

To generate and build a single program in a project, the syntax is as follows:

```
[acupath]\acubench80.exe /build [projectdir]wkspc.pjt prg.psf
```

Using a build script

To generate and build multiple workspaces at the same time, you can create Windows script (batch) files. In the following example, the script builds three programs residing in two workspaces and prints compiler success or failure messages to the screen:

```
@echo off
rem
-------------------------------------------------------------
set bench_80="C:\acucorp\acucbl8xx\acubench\acubench8x"
echo
*************************************************************
echo ... now building program1
%bench_80% /build c:\project1\sample1.pjt program1.psf
IF %errorlevel%==0 ECHO Success.
IF NOT %errorlevel%==0 ECHO Fail.
echo.
echo ... now building program5
%bench_80% /build c:\project1\sample1.pjt program5.psf
IF %errorlevel%==0 ECHO Success.
IF NOT %errorlevel%==0 ECHO Fail.
echo.
echo ... now building sample25
%bench_80% /build c:\demos\demos.pjt sample25.psf
IF %errorlevel%==0 ECHO Success.
IF NOT %errorlevel%==0 ECHO Fail.
echo.
```

Likewise, a script file could be used to build several complete workspaces and create a log file, or send a message to the system administrator when the builds are complete.

3.6 Using Thin Client Technology with AcuConnect

If you want to enjoy the benefits of centralized application maintenance and the performance characteristics of a “thin” architecture, then you can use Micro Focus’s Thin Client technology with AcuConnect®. In a thin client environment, ACUCOBOL-GT programs running on a UNIX, Linux, or Windows NT/2000/2003 server can present a graphical interface on a Windows display host that is running AcuBench. UNIX and Linux users can enjoy the stability and security of their multi-user environments and the graphical nature of Microsoft Windows.

Micro Focus's thin client technology supports the following server platforms: UNIX, Linux, VMS v7.2 or later, HP e3000, Windows NT Server, Windows 2000 Server, and Windows 2003 Server. The following client platforms are supported: Windows XP, Windows NT 4.0 Workstation, Windows 2000 Professional, and Windows Vista.

Thin client operations allow great flexibility in your development environment. For example, assume your source code is stored on a UNIX or Linux application host and protected by a version control system. You check needed files out to a Windows display host running the workbench and edit your source using Code Editor functions. Source is compiled in the workbench and object code is directed to the remote server. Remote debugging capabilities let you test your application, which is displayed on the Windows client while it is running on the UNIX or Linux server. When you finish your work, you check your source files back in to the server.

You use the Build/Use Thin Client menu command to signal the workbench that you want to use thin client functions. Before you compile for thin client, you need to set additional project properties, project settings, and runtime options, and create an alias that contains your thin client command line. You can create object files or libraries in a remote server directory and maintain a UNIX runtime configuration file. Debug your application with the AcuBench integrated debugger or the ACUCOBOL-GT runtime debugger. The commands you now use to develop your project locally (including Compile, Build, Execute, and Debug) help you develop your project in a thin client environment. For information about setting up a remote project, see Chapter 6 in this manual.

If your program is already a graphical application, then you have a little conversion work to perform for thin client operations. If your application is character-based, then you have some development options:

1. Maintain your character-based application on a UNIX or Linux host.
2. Migrate your character-based application to graphical.
3. Develop a new graphical user interface for your UNIX or Linux server-based application.

If you want to maintain your character-based application, you can work with it as is in the workbench with thin client. Note that some display limitations may apply when you choose to maintain your character interface. Certain functions may not appear as expected, and other unsupported functions may not display at all. You may decide to alter your code to work around any unacceptable behavior. Limitations are described in **section 5.1.1** in the *AcuConnect User's Guide*.

If you decide to migrate your character-based program to graphical, Micro Focus's thin client solution can play an important role. Start with the ACUCOBOL-GT Character-to-GUI Wizard, which can convert your character screens to graphical whether or not your program uses a Screen section. Add other screens to your application using the Screen Designer. Include ActiveX controls in your interface as well as ACUCOBOL-GT's standard controls. For information about the Character-to-GUI Wizard, see **Appendix 24, "Appendix A: Bringing Existing Code Into AcuBench,"** in this manual. For more information about Micro Focus's thin client technology, refer to the *AcuConnect User's Guide*.

If you choose to change the look of your interface completely, you can develop a new graphical interface for your application from scratch. The AcuBench graphical Screen Designer provides access to a wide range of graphical technology, including standard and ActiveX controls. The Screen Designer's interface and functions are described in Chapters 13-15 in this manual.

4 Customize Your Working Environment

Key Topics

Introduction 4-2
The Tools/Options Dialog 4-2
Setting Environment Options 4-4
Setting Code Editor Options 4-14
Setting Screen Designer Options 4-19
Setting Code Generator Options 4-19
Setting Data Designer Options 4-27
Setting Report Writer Options 4-28
The Customize Dialog 4-28

4.1 Introduction

Many elements and attributes of the AcuBench interface can be customized to suit your preferences. Workbench windows and toolbars can be positioned directly with the mouse, or hidden and displayed via the View menu or the element's right-click menu. Toolbars can be created or changed and links can be added to external applications (such as a calculator, for example, or a Web browser). Keyboard shortcuts used to invoke workbench commands can be added, modified, and removed. And that's only the beginning.

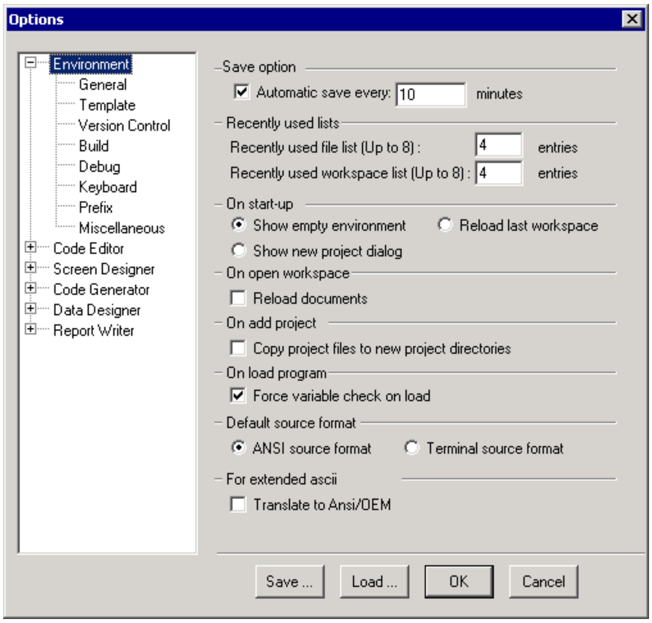
Many aspects of AcuBench's basic functionality can also be customized to adapt the working environment to your preferences. Code generation behaviors, default directory structures, default properties for screens, reports, and controls, and other important workbench features can be adapted to your standards, to suit your needs.

Toolbar creation and editing, and the creation of links to external applications, is handled through the Tools/Customize interface. This interface is discussed in **section 4.9**.

Other customizable aspects of AcuBench can be viewed and modified in the Tools/Options dialog, introduced in **section 4.2** and discussed throughout the greater part of this chapter.

4.2 The Tools/Options Dialog

The Options dialog organizes customizable workbench options into six main categories: Environment, Code Editor, Screen Designer, Code Generator, Data Designer, and Report Writer. Each of these categories is represented as an item in a tree view along the left side of the interface. Each tree can be expanded to reveal sub-categories of information that can be customized for each major category. When you select an item in the tree, options related to that item appear on the right side of the interface.



Save option
[✓] Automatic save every: 10 minutes

Recently used lists
Recently used file list (Up to 8) : 4 entries
Recently used workspace list (Up to 8) : 4 entries

On start-up
[● Show empty environment] ○ Reload last workspace
○ Show new project dialog

On open workspace
[ ] Reload documents

On add project
[ ] Copy project files to new project directories

On load program
[✓] Force variable check on load

Default source format
[● ANSI source format] ○ Terminal source format

For extended ascii
[ ] Translate to Ansi/OEM

Some options displayed here pertain to user interface customization. Others allow you to configure support for such functions as workspace build behavior, Version Control system support, and Code Insight (code completion), as well as to create template code and set screen and report defaults, and more.

It is important that you take the time to get acquainted with all of the options available in the Options dialog.

Using the Tools/Options dialog

To change your workspace options, open the Tools menu and select **Options**. The tree view on the left side of the screen lists each of the major categories that you can customize. Click the plus sign to the left of a category to expand the tree and see all of the available subcategories for that option. Select an item in the tree, then make your changes to the options listed on the right side of the window, according to the information provided in this chapter.

When you are finished making changes, click **OK** to save your changes. More information about different ways to save and reuse your Tools/Options settings appear later in this section.

Saving option settings

Option settings are automatically saved when you click **OK** to close the dialog. They are saved to a file called "AcuBench80.ini", located in your Application Data directory. When AcuBench is launched, the settings are read and applied to the current workspace.

You can save the current settings to a specified file by clicking the **Save** button at the bottom of the dialog. You determine the file name and path; AcuBench automatically assigns the file an extension of ".ini". Because the file contains important default settings for screens and reports, as well as code templates and build options, it can be useful to create a customized ".ini" file to be shared by all members of the development team.

Caution: Do not remove the "AcuBench80.ini" file that resides in the AcuBench installation directory.

Loading option settings

When you change workstations, upgrade to a new version of AcuBench, or make changes to the AcuBench ".ini" file that your team uses for consistency management, you can load your customized ".ini" file at any time. You do not need to have a workspace open to load a new AcuBench ".ini" file.

To load the file:

1. Open the Tools/Options interface and click the **Load** button at the bottom of the screen.
2. Navigate to the directory containing the file, select the file, and click **Open**.
3. Click **OK** to close the Tools/Options interface.

Some changes, like automatic save times and color settings, go into effect immediately. Others, like control property defaults, are applied to new elements added to a workspace, but are not retroactively applied to existing program elements.

4.3 Setting Environment Options

The Environment configuration options are organized into the following subcategories:

General	Workbench startup options
Template	Options for configuring the File/New dialog and an interface for adding file templates
Version Control	An interface for defining version control system commands
Build	Used to determine what actions AcuBench performs before parsing, building/compiling, and executing
Debug	Defines some AcuBench integrated debugger behavior
Keyboard	An interface for assigning keyboard shortcuts to workbench commands and changing existing shortcuts
Prefix	Defines the default folder names and file locations associated with new projects
Miscellaneous	Options that control how the COPYPATH variable is defined and whether an imported source file is automatically converted to an AcuBench program

4.3.1 General Environment Options

The options in the General category allow you to control:

- the timing of the automatic save function
- the Recently Used Files list
- what the workbench loads when it is first started or opens a workspace
- how AcuBench handles adding existing projects to new workspaces
- whether AcuBench validates screen- and report-related variables when a program is loaded into memory
- the default source file format
- the translation of ASCII extended characters to Windows ANSI extended characters

Use the following procedures to set General Environment options, as needed:

1. Select the **General** category of the Environment options.
2. **"Automatic save" option:** To disable the default automatic save setting, clear the **Automatic save** check box. You can change the default time setting by entering a new value (from 1 to 120) in the text box. The automatic save function applies to source files (".cbl"), workspace files (".pjt"), program structure files (".psf"), and data layout files (".dlt"). The automatic save function updates disk files and saves information to the registry, allowing AcuBench to restore your project to a consistent state after an unexpected program interruption.
3. **"Recently used" lists:** To change the size of the "Recently used file list" or the "Recently used workspace list," enter an integer value between 1 and 8 in the corresponding entry field.
4. **"On start-up" options:** In the "On start-up" section, select a radio button to specify what the workbench displays or loads when it is first started.
5. **"On add project" option:** Normally, if you add an existing project to a new workspace (using the "Open Project" command), AcuBench does not move or copy the project's files from their existing location. To change this behavior, and have AcuBench make a copy of the existing project's files within the new workspace's directory structure, select the "Copy project files to new project directories" check box.
6. **"On open workspace" option:** This option determines how AcuBench behaves when a workspace is opened. Enable the check box to direct the workbench to reopen the documents that were open the last time the workspace was open.
7. **"Force variable check on load" option:** If this option is selected, when AcuBench loads a program into memory, it automatically verifies that all variables associated with screen or report controls exist, whether in an AcuBench-generated COPY file or FD, or in an external COPY file. If a variable does not exist, AcuBench creates it in Working-Storage. To bypass the variable check, de-select this option.
8. **"Default source format":** Specify the default source format type by selecting a radio button in the "Default source format" group.

9. **Extended ASCII characters:** Enable **Translate to ANSI/OEM** to cause the workbench to translate extended ASCII characters that originate in the OEM environment to Windows ANSI extended characters. Characters are retranslated back to OEM when the document is saved.

4.3.2 Template Options

When you work in AcuBench, you are provided with a series of default templates, used as the basis for creating new programs, data layout files, screens, reports, and files. In addition to these defaults, you have the option to create your own custom screen, report, and file templates. Using custom templates can help you to maintain a consistent code structure and program appearance across an application or application suite.

The Template portion of the Tools/Options interface lets you determine how both the AcuBench default templates and your own custom templates are made available for your use. You can even entirely disable the File/New dialog that allows you to choose the base template for programs, data layout files, screens, reports, and files, such that basic, default characteristics are always used when you create a new program item.

The process of creating templates is discussed elsewhere in this manual. Source file templates are discussed in **section 12.5.7**, screen templates are discussed in **section 13.4**, and report templates are described in **section 16.4**. All files added using this dialog are displayed in the New File, New Screen, or New Report template group, depending on the template type.

Follow these steps to set Template options:

1. Select the **Template** category of the Environment options.

2. **"Use File/New" options:** Toggle the check boxes in the "Use New/File dialog when creating" group to control when the File/New dialog is displayed. By default, these are all selected, indicating that a File/New interface containing the available AcuBench templates is displayed whenever you create a new program element.

When you de-select an option, the File/New dialog does not appear when you opt to create a new element of the specified type. Instead, a default element type is automatically created. The defaults are:

Program	Standard Program (a program structure file with an empty screen)
Screen	Blank Graphical (an empty graphical screen)
Report	Blank Graphical (an empty graphical report)
File	Source Template (a very basic file with COBOL divisions and AcuBench tags)
FD/SL	Blank (launches the File Designer, bypassing the Import option)

3. **"Customize Template" interface:** To add templates that you have created to the File/New dialog, first select the type of template from the "Template for" drop-down box.

    a. Click **Add** and provide a descriptive title for the template that you want to use.

    b. Click the browse button to navigate to the location of the template file, then select the file and click **Open**.

    c. Add a detailed description of the template in the large entry field at the bottom of the Add New Template File interface.

    d. Click **OK**. When you exit the Tools/Options interface, you can use the File/New command to verify that your template has been added to the appropriate interface.

    To remove a template that you have previously added to the interface, select the template icon and click **Delete**. To modify the title, file path, or description of a template, select its icon and click **Modify**.

    You cannot change the set of default templates using this interface.

4.3.3 Version Control Options

The Version Control options allow you to define commands to work with your version control system. A complete description of the Version Control interface is contained in **Chapter 5, "Chapter 5: Version Control."**

4.3.4 Build Options

The Build options allow you to control some aspects of how the workbench behaves when you run tools, compile programs, execute object files, or build/rebuild the workspace.

Use the following procedures to set Build options:

1. Select the **Build** category of the Environment options.
2. **"When running tools" option:** Enable **Always save all documents** to automatically save all open documents before performing such actions as reparsing, generating code, or compiling, among others.
3. For the three options under **"When compiling programs that include generated source code"**:
    - Enable **Always generate code if program structure file is modified** to generate code before compiling any time that the program structure file has been modified since the last generate action.
    - Enable **Prompt before generating source code** to direct the workbench to always display a confirmation dialog before starting code generation.
    - Enable **Jump to Event Editor when browsing** to direct AcuBench to open the Event Editor when you click on a compilation error message caused by code in the program's ".evt" COPY file.
4. For the two options under **"When executing source (.cbl) or object files"**:
    - Enable **Always re-compile if the original source file is modified** to direct the workbench to recompile a modified source file before executing the associated program.
    - Enable **Prompt before re-compiling source file** if you want the workbench to display a prompt that asks if you want to recompile a modified source file.
5. For the two options under **"When building or reparsing the workspace"**:

- Enable **Always generate modified program (.psf) files** to direct AcuBench to first generate code for all programs (".psf" files) in the workspace that have been modified since the last build/rebuild or reparse, before starting the new build/rebuild or reparse.
- Enable **Always generate modified FD/SL (.dlt) file** to direct the workbench to generate new FD and SL files and update any user-modified file-handling code for all data layout files (".dlt") that have been modified since the last build/rebuild or reparse, before starting the new build/rebuild or reparse.

4.3.5 Debug Options

The Debug options allow you to specify whether AcuBench should perform the same save and code generation actions, prior to starting a program in the debugger, that are performed prior to a build/rebuild. Because you have the option to modify code in the debugger, this option can provide a handy way to speed up the testing of new changes. Consider the following case:

- In the Tools/Options/Build interface, you have specified that AcuBench should generate code before compiling or recompiling source code.
- In the Tools/Options/Debug interface, you have specified that the integrated debugger should follow build options.
- As you are debugging a program in the integrated debugger, you make some changes to your event code.
- When you exit and re-enter the debugger, AcuBench checks for changes, regenerates your program, compiles, and restarts the debug process. You can immediately test your changes to the code.

There is an important caveat for those considering this option. If you have selected "Always save all documents" in the Tools/Options/Build interface, having the debugger use the build options may have a significant impact on performance. Because saving a file changes its timestamp, this combination of options causes AcuBench to perform a detailed check for code changes in every file every time you enter the debugger.

This interface also lets you determine whether the current line of execution in the debugger is highlighted and what the highlight color is.

To set Debug options, perform the following steps, as needed:

1. Select the **Debug** category of the Environment options.
2. Enable the **Follow the options specified in 'Build' page** check box to direct the workbench to apply the settings in the Build category to the set of actions taken prior to starting a program in the debugger.
3. Enable the **Colorize line of execution** check box if you want the current line of execution in your code to be highlighted. Choose a highlight color in the drop-down box to the right of the check box.

4.3.6 Keyboard Options

The Keyboard options allow you to view, define, redefine, or remove the keyboard shortcut keys associated with workbench commands. A complete list of workbench commands and their default keyboard shortcut settings is provided in Chapter 23.

At the top of the Keyboard options screen, next to the Categories label, is a drop-down list of three command categories: Main, Code Editor, and Screen Designer. Main is a superset of Code Editor and Screen Designer.

You can set Keyboard options as follows:

1. Select the **Keyboard** category of the Environment options.
2. Select the category of commands that you want to view or modify.

To view a shortcut key definition, locate the command of interest in the list. The shortcut key definition, if any, is displayed in the column to the right of the command.

Use the following procedures to assign a shortcut key:

1. Locate the command of interest in the list and click on it. Note that a short description of the command is displayed to the right of the "Currently assigned to" label near the bottom of the screen.

2. Click in the **Shortcut key** entry field.
3. Enter the desired key or key combination by pressing keys on the keyboard. As you press each key, the keystroke appears in the entry field.

   Look under the "Currently assigned to" label to see if that key (or key combination) is already defined for another command. It is possible to assign the same key to more than one command. However, doing so is only desirable when the commands can only be activated in mutually exclusive states. For example, the key combination "Ctrl+G" might be defined in the Code Editor to perform a GotoLine command and in the Screen Designer to toggle the grid (ToggleGrid). Because you can't be actively working in both the Code Editor and Screen Designer at the same time, typing "Ctrl+G" is never ambiguous, and the correct action is always performed. In the case where the commands do not belong to mutually exclusive states, the results are undefined.
4. Click **Assign**.

To remove a shortcut key, locate the desired command and click on it. Note that a short description of the command is displayed to the right of the "Currently assigned to" label. Click **Remove**.

To restore the default shortcut definitions, click **Reset All**.

4.3.7 Prefix Options

The Prefix options allow you to determine the naming convention used to provide default names for new projects and files and the default directory paths established for those elements. These default directory paths are used to create a directory structure for each new project that you create. You can change this structure only when you are creating a new project, when you click the More Info button. Once a project has been created, the directory structure can no longer be changed.

Use the following procedures to set your default Prefix options:

1. Select the **Prefix** category of the Environment options.
2. **File prefix fields:** For each file type, enter the string to be used as the default base name for creating new files of that type.

3. **Working directory fields:** For each working directory, enter the desired folder name in the entry field.

Each of the working directory entries in this interface has a corresponding macro, which you can use as a shorthand method of referring to the directory path in other AcuBench interfaces, such as the Project Settings interface. These macros include the following:

Macro	**Definition**
%sourcedir%	The path assigned in the Source directory field
%screendir%	The path assigned in the Screen directory field
%reportdir%	The path assigned in the Report directory field
%copylib%	The path assigned in the Copylib directory field
%objectdir%	The path assigned in the Object directory field
%listdir%	The path assigned in the List directory field
%resdir%	The path assigned in the Resource directory field
%layoutdir%	The path assigned in the FD directory field

If you are using the default project directory structure, the following syntax:

```
.\object\@.acu
```

is functionally equivalent to:

```
%objectdir%\@.acu
```

Note that, by default, these working directory macros, rather than full directory paths, are used in COPYPATH definition that appears on the Environment tab of the Project Settings window.

4.3.8 Miscellaneous Environment Options

Although somewhat obscured by their placement in this category, the Miscellaneous options include three very important options that affect:

- how the COPYPATH environment variable is defined
- whether when a source file is added to a project, tags are inserted in the file and a program structure file is created

The definition of the COPYPATH environment variable determines where AcuBench looks for COPY files and whether and which files are found. Its correct definition is essential to a successful build. For a detailed discussion of the COPYPATH environment variable, see **section 7.5.2**.

Use the following steps to set Miscellaneous options, as needed:

1. Select the **Miscellaneous** category of the Environment options.
2. For the two options under **COPYPATH**:
    - Enable **When adding a COPY file** to automatically modify the definition of COPYPATH to include the path to the new COPY file. COPYPATH is defined on the Environment tab of the Project Settings window.
    - Enable **When modifying COPY path in Add/Remove files dialog** to modify the definition of COPYPATH when it is changed via the Add/Remove Files dialog.
3. Under the **"Automatically create a program structure file"** option, enable **When adding a source file to the project's source folder** to direct AcuBench to create a program structure file for any source file that is added to a project. As a part of this process, the source file is modified to include a set of AcuBench tags (see **section 3.3**). Note that only programs that have a program structure file are able to use the Screen Designer and other tools that generate code.

    Keep in mind that in the Add Program dialog, you can specify, on an individual basis, whether a program structure file should be created for any given program being added. For an introduction to AcuBench programs, see **Chapter 3, section 3.2.3, "Programs."** More detail is available in **Chapter 9, "Chapter 9: Working with Programs."** For a discussion of adding source files to a project, see **section 9.3.2**.

4.4 Setting Code Editor Options

The Code Editor portion of the Tools/Options dialog provides interfaces for customizing both the appearance and the functionality of the Code Editor. This includes options that determine the behavior of Code Editor tools, including the Code Completion pop-up window and the AcuBench paragraph, variable, constant, and COPY file lists.

4.4.1 Code Editor General, Format, Tabs, and Keyword Options

General options for the Code Editor let you determine Line Number pane width, Code Editor line length, ASCII code settings, and behavior of the editor's vertical block select function.

Format options can be used to change the colors that identify each ANSI display field (Sequence Number area, Indicator area, area A, area B, and Identification area). You can also customize the appearance of code text types (Comment, Keyword, String, Number, and Text) and designate the text string that is inserted in ANSI format columns 73-80 when you modify source code.

Format options also allow you to change the default column settings of your file. The default settings are the ANSI-specified standard settings. If you are working with terminal source format, you may want to change these default settings. An explanation of terminal source format is found in Book 1 of the ACUCOBOL-GT documentation set.

Tabs options allow you to direct that the editor support the tabs in your source code or automatically convert the tabs in your code to spaces. You can also set up to 32 custom tab stops.

Finally, Keyword options give you the ability to add, delete, and modify keyword sets. You can also determine how far the next line after the keyword will be indented, if at all.

Note that these keyword settings affect only the workbench's colorization and auto-indent functions. Adding or deleting a keyword in a default list does not affect the behavior of the ACUCOBOL-GT compiler. To change the list of reserved words recognized by the compiler, use the "-R" flag, described in Book 1 of the ACUCOBOL-GT documentation set.

The Code Editor General, Format, Tabs, and Keyword options are described in **Chapter 11, "Chapter 11: Configuring the Code Editor."**

4.4.2 Code Insight Options

The Code Insight functions are intended to facilitate source code creation, especially when working with graphical screens and reports. The code parameter feature displays hints (similar to the tool tips that appear when the mouse hovers over a button on a toolbar) for using certain COBOL verbs. Code completion provides pop-up lists of options to help complete some code statements and internal library routine calls. Code parameters and code completion are available for the following verbs: CALL, PERFORM, READ, WRITE, START, DELETE, REWRITE, OPEN, CLOSE, CHAIN, MODIFY, INQUIRE, DESTROY, and GO. For more information about using these functions, see **Chapter 12, section 12.5.4, "Using Code Insight Functions."**

With code templates, you can create a library of commonly used code paragraphs or sections. This list of templates can be accessed from within the Code Editor using the hotkey **Ctrl+J**.

You can set Code Insight options using the following procedures:

1. Select the **Code Insight** category of the Code Editor options.
2. Select the **Code Completion** feature if you want the workbench to display a list of possibilities for completing your code statement.
3. Select **Code Parameters** if you want AcuBench to display a pop-up hint that generally describes the function of the verb you have entered.

4. Use the “Code Template” text box to define frequently used code as a template that can be inserted into your source code with a mouse click. Approximately 20 default templates are available in the workbench. These templates may be modified directly in the dialog’s “Code” window.

   - Click **New** to add a new template to the list, then enter the code for the new template in the “Code” window. You can either paste code from the Code Editor or another source, or type directly into the window.

   - Add a pipe sign (“|”) in your code to position the cursor when the template is used. If you include multiple pipe signs, the cursor appears at the first pipe sign in the last line of the code template text. Use a double pipe sign to indicate that the following text should be inserted as a comment.

   - You can display the current list of code templates when working in the Code Editor by pressing Ctrl+J or selecting the Edit/Advanced/Code Template command.

4.4.3 Paragraph List, Variable List, Constant List, and COPY File List Options

The Paragraph List, Variable List, Constant List, and COPY File List options allow you to specify how each of these pop-up lists is sorted, and to determine which columns of information are displayed in the box. Because each of these lists functions in a similar manner, the configuration options for each list are likewise similar.

You can set the options Paragraph, Variable, Constant, and COPY File lists as follows:

1. Select the appropriate category from the Code Editor options.

2. In the **“Sort By”** area, select one of the available radio buttons to determine how the pop-up list will be sorted.

   To sort items in order of their occurrence in the program, mark the **Occurrence in the program** radio button.

To list items in alphabetical order, choose the **Alphabetically** radio button.

The Paragraph List has the following unique options:

- To sort COPY file names alphabetically and have paragraph names sorted by occurrence within each COPY file, choose **Occurrence within alphabetic sort of copybooks**.
- To sort COPY file names alphabetically and have paragraph names sorted alphabetically within them, select **Alphabetically within alphabetical sort of copybooks**.

The Variable List has the following unique options:

- To sort variable names by level number, choose the **Level number** radio button.
- To sort variables by picture clause, select **Picture description**.

The Constant List has the following unique option:

- To sort constants by value, select **Value**.

The COPY Files List has the following unique options:

- To sort COPY files alphabetically by directory, select the **Directory** radio button.
- To sort COPY files based on the names of the files in which they are declared, choose **Declared file**.

3. In the **"Show" option** area, mark one or more of the available check boxes to determine what information will appear in the pop-up list.

   The Paragraph, Variable, and Constant lists all have the option to display the name of the COPY file in which the program element is declared. To select this option, mark **The name of the copybook**.

   The Variable List has the following additional options:

   - To display level numbers, select **Level number**.
   - To display picture clauses, select **Picture description**.
   - If you want to see FILLER data items in the Variable List, mark **Include FILLER data item**, as well.

The Constant List has the following additional option:

- Select **Value** to see each constant's value in the Constant list.

Finally, the COPY File List has the following options:

- To display each COPY file's directory, select the **Directory** check box.
- To see the file in which each COPY file is declared, select **Declared file**.

4.5 Setting Screen Designer Options

You can set Screen Designer features, including grid behavior, control property default values, and control property visibility options in the Screen Designer portion of the Options dialog box.

These options are discussed in detail in **Chapter 13, "Chapter 13: Configuring the Screen Designer."**

4.6 Setting Code Generator Options

The Code Generator options allow you to control which files are generated and, in some cases, what parts of a file are generated when you initiate a Generate action. You can also determine the default program tags you want AcuBench to generate, including "extern" tags, which allow the automatic generation of ".def" and COPY file statements for external files. For a detailed discussion of workbench code generation facilities, see **section 3.3**.

4.6.1 Generate Document Options

The "Generate Document" options let you determine the automatic code generation behavior you want AcuBench to perform. You can also change the default file extensions for generated files in this interface. When you change a default file extension, that change is also reflected in the Program

Properties dialog. Note that changes to these settings made on the Code Generation tab of the Program Properties dialog override the default settings made in this Options/Generate Document interface. Refer to **section 9.2.2, "AcuBench Program Properties,"** for information about this dialog.

Set any desired Generate Document options as follows:

1. Select the **Generate Document** category of the Code Generator options.
2. The list of options under **"Generated documents"** are selected by default. These options have the following functions:
   - When **Program file** is selected, and **Regenerate tagged area only** is *not* selected, AcuBench generates code for the program source file, replacing the existing *program_name.cbl* file. You can change the default file extension (".cbl") in the associated entry field.
   - When both **Program file** and **Regenerate tagged area only** are selected, AcuBench generates code in the tagged portions of the program file (".cbl") only. This preserves any code that you may have added outside of the AcuBench tags.
   - When **Working storage** is selected, AcuBench generates the Working-Storage COPY file (".wrk"). You can change the default file extension in the associated entry field.
   - When both **Working storage** and **Exclude variables in program file** are selected, AcuBench does *not* generate Working-Storage items in the ".wrk" file for variables that are already declared in the program file (".cbl").
   - When **Event paragraph** is selected, AcuBench generates the event paragraph COPY file (".evt"). You can change the default file extension in the associated entry field.
   - When both **Event paragraph** and **Exclude paragraphs in program file** are selected, AcuBench does *not* generate paragraphs that are already included in the program file. The workbench assumes that a paragraph in the program file with the same name as a paragraph that could be generated in the ".evt" file suppresses the paragraph that could be generated.
   - When **Screen section** is selected, the workbench generates a Screen Section COPY file (".scr"). You can change the default file extension in the associated entry field.

- When **Report section** is selected, the workbench generates a report COPY file (".rpt"). You can change the default file extension in the associated entry field.
- When **Procedure division** is selected, AcuBench generates a Procedure Division COPY file (".prd"). You can change the default file extension in the associated entry field.
- When **Menu paragraph** is selected, AcuBench generates a menu paragraph COPY file (".mnu"). You can change the default file extension in the associated entry field.
- When **Linkage section** is selected, AcuBench generates a Linkage section COPY file (".lks"). You can change the default file extension in the associated entry field.

3. For the two **"Attributes of generated files"** options:

   - Select **Program file as read-only** to cause the source program file (".cbl") to be created as a read-only file.
   - Select **Copy book as read-only** to cause all generated COPY files to be created as read-only files. This is the recommended behavior.

4. **"Screen Control Focus" option:** Set a value for the ACCEPT-CONTROL variable (from 0-4) in the **ACCEPT-CONTROL value** entry field. The default value is "1". If you do not want an ACCEPT-CONTROL statement generated at all, select **(none)**. For information about the purpose of ACCEPT-CONTROL, see General Rule 4 of section 4.2.3, "Special-Names Paragraph," in *ACUCOBOL-GT Reference Manual*.

5. **"Generated document format" options:** Set the **Merge to one program file** radio button if you want all your AcuBench-generated code, including COPY files, written to a single program file. Note that AcuBench can handle approximately 200,000 total lines. If your files total more than this amount, the merged file may fail to produce.

   Set the **Split to multiple copy files** radio button if you want AcuBench to maintain your COPY files in separate files.

6. **“Attributes of Data Set code generation” option:** Select **Link file close to open** if you want AcuBench to automatically generate a Close I/O paragraph for a data set when the Open option is already set. This code generation option is especially helpful if you are upgrading your Version 5.*x* AcuBench project to Version 6.*x* or later.

4.6.2 Program Tag Options

To support automated code generation, if you choose, AcuBench places special comment tags in your COBOL source file. These tags reserve areas of your program for code generated by AcuBench. For example, if you use the Screen Designer to create a screen for your program, every time you generate code for the program, the workbench creates a COPY file that describes the screen, and a COPY statement is inserted between the copy-screen tags in your program’s Screen section. You can control whether specific tag pairs are automatically generated or suppressed via the Tools/Options/Code Generator/Program Tag interface.

AcuBench tags always have the format:

```
*{Bench}tag-name
*{Bench}end
```

When working with programs that include AcuBench-generated code, always be sure to add any user-defined code *outside* of the “{Bench}” tags. Everything between the tags is deleted and re-created every time you generate the program, but code outside the tags is preserved (unless you have specifically elected otherwise in your Code Generator options).

If you were to use all of AcuBench’s facilities for automatically generating code, your program could have the following set of tags in your source file (“.cbl”):

```
*{Bench}prg-comment
* Demo.cbl
* Demo.cbl is generated from C:\Sample\Demo.Psf
*{Bench}end
 IDENTIFICATION              DIVISION.
*{Bench}prgid
 PROGRAM-ID. Demo.
 AUTHOR. ljones.
 DATE-WRITTEN. Wednesday, January 04, 2006 9:54:03 AM.
 REMARKS.
*{Bench}end
```

Program description comments and program ID information is pulled from settings in the Project/Properties dialog.

A Special Names definition COPY file is included whenever ActiveX controls are used in a screen created in the Screen Designer:

```
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL NAMES.
*{Bench}activex-def
 COPY "MSChart.def".
 COPY "Acuclass.Def".
     .
*{Bench}end
```

In the example above, the program uses the MSChart ActiveX control, so AcuBench has added references to two required definition files between the “activex-def” {Bench} tags.

If in your Windows environment you have “,” (comma) defined as the decimal symbol, a DECIMAL POINT IS COMMA statement is inserted between the “decimal-point” tags, which the code generator places directly after any ActiveX tags in the Special Names section.

```
*{Bench}decimal-point
*{Bench}end
```

COPY statements to include SELECT statement (“.sl”) COPY files generated by the File Designer are inserted between the “file-control” tags:

```
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
*{Bench}file-control
 COPY Demo.SL
*{Bench}end
```

COPY statements to include file descriptor (“.fd”) COPY files generated by the File Designer are inserted between the “file” tags:

```
 DATA DIVISION.
 FILE SECTION.
*{Bench}file
 COPY Demo.FD
*{Bench}end
```

COPY statements to include standard ACUCOBOL-GT definition (".def") COPY files are inserted between the "acu-def" tags:

```
 WORKING-STORAGE SECTION.
*{Bench}acu-def
 COPY "acugui.def".
 COPY "acucobol.def".
 COPY "crtvars.def".
 COPY "activex.def".
 COPY "showmsg.def".
*{Bench}end
```

COPY statements to include Working-Storage (".wrk") COPY files containing items defined by the graphical design tools are inserted between the "copy-working" tags:

```
*{Bench}copy-working
 COPY Demo.wrk
*{Bench}end
```

COPY statements to include external ".def" files are inserted between the "extern-def" tags:

```
*{Bench}extern-def
 COPY externalfile.def
*{Bench}end
```

COPY statements to include Linkage section (".lks") COPY files containing items defined in the Linkage Editor are inserted between the "linkage" tags:

```
 LINKAGE SECTION.
*{Bench}linkage
 COPY Demo.lks
*{Bench}end
```

COPY statements to include screen (".scr") COPY files created by the Screen Designer are inserted between the "copy-screen" tags:

```
 SCREEN SECTION.
*{Bench}copy-screen
 COPY MainMenu.scr
 COPY Orders.scr
 COPY Invoices.scr
*{Bench}end
```

A Procedure Division header with a USING phrase is generated between the “linkpara” tags when linkage items are defined in the Linkage Editor:

```
*{Bench}linkpara
 PROCEDURE DIVISION USING order-id.
*{Bench}end
```

A declarative section with code to handle standard file error conditions is inserted between the “declarative” tags when data sets are defined with the Data Set Designer. You can also control what is generated inside the “declarative” tags via the Event Editor. Refer to **Chapter 14, section 14.8, “Associating Code with Screen Elements.”** for more information:

```
*{Bench}declarative
 DECLARATIVES.
 INPUT-ERROR SECTION.
     USE AFTER STANDARD ERROR PROCEDURE ON INPUT.
 ...
 END DECLARATIVES.
*{Bench}end
```

An insertion point for user-defined code to be executed before the program’s initial routine is added at the start of the “Acu-Main-Logic” paragraph. When “Before Program” code is defined in the Event Editor, AcuBench generates the necessary PERFORM statement between the tags.

```
*{Bench}entry-befprg
     PERFORM Before-Orders-Program
*{Bench}end
```

Code to call the program’s main screen (a paragraph included in the “.prd” file generated by the Screen Designer) is inserted between the “run-mainscr” tags:

```
*{Bench}run-mainscr
     PERFORM Acu-MainMenu-Routine
*{Bench}end
```

COPY statements that include procedure, event, and menu (“.prd”, “.evt”, “.mnu”) COPY files are inserted between the “copy-procedure” tags:

```
*{Bench}copy-procedure
 COPY Demo.prd
 COPY Demo.mnu
 COPY Demo.evt
*{Bench}end
```

COPY statements to include corporate external Procedure Division COPY files are inserted between the “extern-cpy” tags:

```
*{Bench}extern-cpy
 COPY externalcopyfile.cpy
*{Bench}end
```

If you want to view or print a report, a call to the master print paragraph is inserted between these tags:

```
*{Bench}reportname-masterprintpara
*{Bench}end
```

Including and excluding tags

Tags are included in a new program when you generate the program the first time. If you use the Code Generator/Program Tag section of the Tools/Options interface to indicate that you do not want certain tags created, then later change your mind, the missing tags will not be added to programs that you have previously generated. You can add the tags manually, after which AcuBench will generate any corresponding code between the tags.

Tags are inserted into an existing program if you specify that AcuBench should create a program structure file when you add the program to the project. For more information about creating a program structure file when adding a program, see **section 24.3, “Creating a PSF for an Existing Program.”** As a general rule, the workbench inserts tags at the end of each relevant section (such as the Screen section). If the workbench cannot determine where the end of a section is, it does not insert *any* tags.

In addition to the typical range of tags, you can also direct the workbench to generate “extern” tags, which contain COPY statements for external “.def” and COPY files. Note that standard “.def” and COPY file statements are generated inside “acu-def” and “copy-procedure” tags.

You can set Program Tag options as follows:

1. Select the **Program Tag** category of the Code Generator options.
2. Set the **Default Tag Name** check boxes for the tag pairs that you want AcuBench to generate automatically. Clear check boxes for the tag pairs you don’t want generated.

Note that the selected tags are added to new AcuBench programs, or to programs for which a new program structure file is being created. If tags are added or removed from the list in the Tools/Options interface, no change occurs in existing AcuBench programs. If you manually add an appropriate set of tags to an existing program, the code generation facility will generate any corresponding code between those tags.

3. Select **Use External Working Storage Copy File Tag** if you want the workbench to generate tags for an external COPY file. These tags are generated into the ".cbl" file, in the Working-Storage section, just after the "{Bench}acu-def" tags.

   Enter the file name in the associated **Copy File List** box, using the **Browse** button to navigate to the desired COPY file. Use the **Delete** and **Delete All** buttons to remove files from the list.

4. Select **Use External Procedure Division Copy File Tag** if you want AcuBench to generate tags for a COPY file. These tags are generated in the ".cbl" file, in the Procedure Division, just after the "{Bench}copy-procedure" tags.

   Enter the COPY file name in the associated **Copy File List** box, using the **Browse** button to navigate to the desired COPY file. Use the **Delete** and **Delete All** buttons to remove files from the list.

4.7 Setting Data Designer Options

The Data Designer configuration options allow you to specify the inclusion or exclusion of comments in files created by the data design tools (File Designer, Working Storage Editor, and Linkage Editor). In addition, you can select the colors used to display "linked" items (items brought in via a COPY statement) included with the data designers. This interface also lets you determine level-number interval settings for the graphical file designers.

To set General Data Designer options, use the following steps:

1. Select the **General** category of the Data Designer options.
2. **"Comment" options:** Select a radio button to specify the inclusion or exclusion of comments in files created by the data design tools.

3. Enter a default prefix to be applied to any new I/O paragraphs in the **Function prefix** entry field.

4. **"Copy file color" options:** Specify the colors used to show "linked" data items in a data structure (included via the "Link to file" option of the Add Item list). Linked items may themselves include items via COPY statements. Items included as the result of nested COPY statements are also shown in a contrasting color. You can set the color used at each level by double-clicking in the Text Color field of the desired level and selecting a color.

You can set Graphical FD, WS, and Linkage Data Designer options with the following procedures:

1. Select the **Graphical FD**, **Graphical WS**, or **Graphical Linkage** category of the Data Designer options.

2. If desired, change the default element prefix in the Element Prefix entry field.

3. Set the level-number intervals for the second, third, and fourth sub-item levels for code generated by the File Designer. You cannot change the first level setting.

4. Enter an interval for any levels past the fourth level. Your level-number sequence, along with a preview of element names using the prefix convention you have chosen, is shown in the display box.

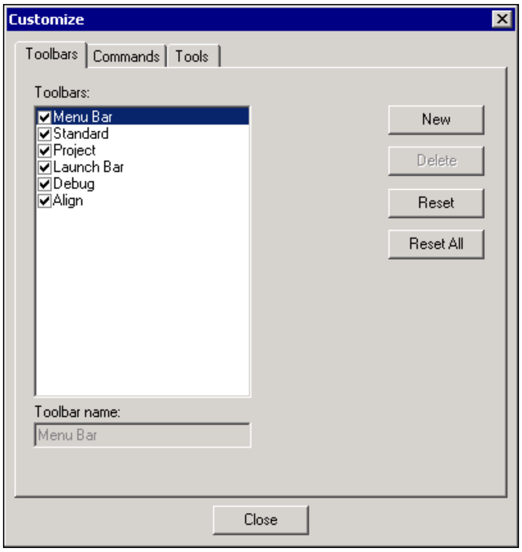
4.8 Setting Report Writer Options

You can set Report Composer features, including grid behavior, report control property default values, and report control property visibility options in the Report Writer portion of the Options dialog box.

These options are discussed in **Chapter 16, "Chapter 16: Configuring the Report Composer."**

4.9 The Customize Dialog

The Tools/Customize dialog can be used to modify any of the default AcuBench toolbars, create custom toolbars for AcuBench, or establish links to external applications. This dialog contains three tabs.



The Toolbars tab allows you to create, delete, or reset a toolbar. It also provides an alternate mechanism for controlling which toolbars are hidden or displayed.

The Commands tab lists all of the commands in AcuBench that can be placed on a toolbar, including the links you've established to external applications (listed in the Launch Tools category).

The Tools tab hosts the interface that allows you to establish and configure links to external applications.

4.9.1 Customizing AcuBench Toolbars

One method through which the AcuBench workspace can be adapted to your individual patterns of use is through the ability to customize existing toolbars and create new ones.

If you have modified a toolbar and want to return to the default settings, you can select a toolbar on the Toolbars tab of the Customize window and select **Reset**. To return all toolbars to their default settings, select **Reset All**.

Selecting a toolbar name in the “Toolbars” list makes it visible on the screen; de-selecting a toolbar name hides that toolbar.

You can delete any toolbar that you have created by selecting the toolbar name in the “Toolbars” list and clicking **Delete**. You cannot delete any of the default, AcuBench toolbars.

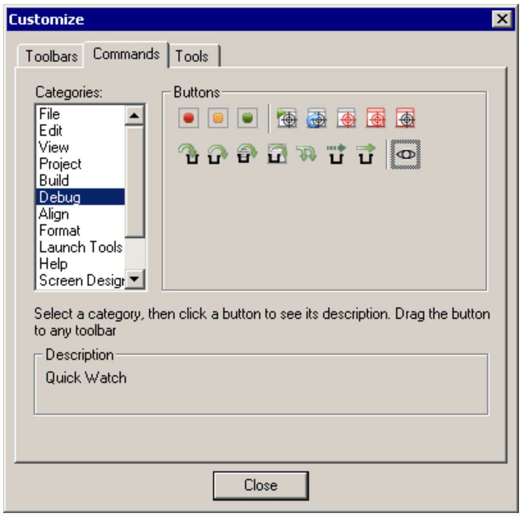
To create a new toolbar

1. Select **Customize** from the Tools menu. The dialog opens with the **Toolbars** tab selected.
2. Click **New**. A “New Toolbar” dialog opens.
3. Enter a descriptive name for your new toolbar, then click **OK**.

   The name of your new toolbar is added to the list of toolbars in the dialog, and a small, floating toolbar appears just below the toolbar area of the workspace.
4. Follow the instructions below to build your toolbar. As with any other toolbar, you can dock your custom toolbar(s) at the top of the screen, or in a variety of other locations in the workspace.

To add buttons to a toolbar

1. Select the **Commands** tab of the Customize dialog.



Select a category, then click a button to see its description. Drag the button to any toolbar

Description

Quick Watch

2. Select a type of command from the "Categories" list box on the left side of the screen. A list of icons associated with the category that you have selected appears in the "Buttons" frame on the right of the screen.

3. Select a button from the list to see a description of the associated functionality in the "Description" frame.

4. Drag the button from the "Buttons" frame to a toolbar to add that command to the toolbar.

5. When you are finished using the Customize dialog, click **Close**.

To remove buttons from a toolbar

1. Make sure that you are in the **Commands** tab of the Customize dialog.

2. Drag the button from the toolbar to any part of the Customize dialog.

To add or remove spacers from a toolbar

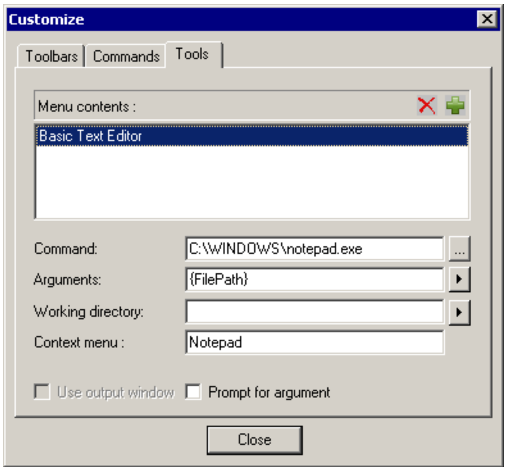
1. Make sure that you are in the **Commands** tab of the Customize dialog.

2. To add a spacer character to the left of an existing button, click the button and drag it slightly to the right, then release the mouse button.

3. To remove a spacer character from a toolbar, click button to the right of the character and drag it slightly to the left (overlapping the spacer character), then release the mouse button.

4.9.2 Accessing External Applications

When you are working in AcuBench, it may be useful to have easy access to external applications, like Windows Explorer, or a calculator. The Tools tab of the Customize interface allows you to create a link to such external applications. Once the link has been made, you can launch the associated application from a menu within the workbench, or by adding a launch button to any toolbar.



Basic Text Editor

Command: C:\WINDOWS\notepad.exe

Arguments: {FilePath}

Working directory:

Context menu : Notepad

Use output window

Prompt for argument

Add a link to an external application as described below:

1. Click on the **New** button in the upper right of the “Menu contents” list box.

2. In the empty field created for the new list item, enter the name that you want to associate with the application. This is the name that is added to the Tools/Launch Tools menu and the text of the “tool tip” that is displayed when the mouse pointer is held over the command icon on the toolbar.

3. Enter the full path of the external executable that you want to access from within AcuBench in the “Commands” field, or use the browse button to the right of the field to navigate to the executable.

4. The “Arguments” field is used to specify switches or parameters to apply to the executable each time the application is started. You have the choice to leave this blank or enter any options that you want to pass.

   The arrow button to the right of the field expands to provide the following set of default arguments:

Argument	**Description**
File Path {FilePath}	The full path name of the files that are contained in the current project or folder.
File Directory {FileDir}	The directory for the files contained in the current project or folder.
File Name {FileName}	The name of the selected file or files.
Current Directory {CurDir}	The current directory containing the AcuBench application.
Workspace Path {WkspPath}	The full path name for the current workspace.
Project Directory {ProjectDir}	The directory containing the current project.
Project Name {ProjectName}	The name of the current project.

   The %*directory_code*% AcuBench macros can also be used in this interface.

5. If applicable, enter the full path to the directory that the application will use as its working directory in the “Working Directory” field.

By default, the working directory is the same as the location of the application. Three of the tokens used in the Arguments field can also appear in the working directory field. These are {FileDir}, {CurDir}, and {ProjectDir}.

6. Click **Close** to save your changes and exit the Customize interface.

   Note that text in the “Context menu” field is not used at this time.

5 Version Control

Key Topics

Version Control Overview 5-2
The Version Control Interface 5-5
Adding Commands to the Menu Command List 5-8
Modifying the Command List 5-9
Saving the Command List 5-9
Command Variables 5-10
Issuing Version Control Commands 5-11

5.1 Version Control Overview

AcuBench provides support for third party version control systems that include a Windows command-line interface.

Version (source) control systems help developers manage and maintain projects by creating a database of project files and resources, which the version control system monitors and protects. In a team environment, this is essential to allow multiple developers to work in the same project without overwriting or otherwise interfering with one another's work. Developers view and modify project files and resources locally, then return their changes to the version control system. The system logs changes so that prior versions of an item are easily retrieved or restored, and integrates each developer's changes with changes made by the rest of the team.

Most version control systems support the following basic actions:

- Create project
- Create subproject
- Add files to a project
- Check files out of a project
- Update local files with other developers' logged changes
- Check revised files into a project
- Retrieve version history for a project and its files

Version control systems perform a number of useful functions. Their automatic logging facilities can be useful in creating a complete and detailed project history. Version control can also be used to help with consistency management (maintaining a consistent look and feel between modules in an application), and make it easy to branch and merge code (allowing testing of an upcoming release, for example, without affecting the code base for the current release). Version control systems do not take the place of good planning and communication, but they can make managing team contributions to an application significantly easier.

5.1.1 Working with version control in AcuBench

The AcuBench interface to version control software allows you to define commands that call your version control system directly or that execute your version control batch files. These commands are placed on the Version Control menu item that is accessed when you right-click on a project or file node in the Workspace window.

When you select and run a version control command, AcuBench passes the defined command string and arguments to the Windows command shell for execution. By default, the command is echoed to the AcuBench Output window, which is also used to display console messages from the version control system.

Note: If your version control system does not have a Windows command line interface, it cannot be accessed through AcuBench.

When adding version control commands to the AcuBench interface, please keep in mind that there are some commands that are better issued from outside AcuBench, or after AcuBench has been shut down. For example, changes to a particular program or file can easily be committed to the version control repository from within AcuBench. But a project commit (writing all changes made to a project to the version control repository) is better issued from the command line, after AcuBench has been closed, to ensure that all changes to the workspace project file (".pjt") have been written to disk.

5.1.2 Preliminary considerations

As you define your projects, instead of trying to group all of your programs into a single, large workspace, create logical groupings of related programs. Because the AcuBench project and workspace concepts are entirely virtual, these groupings do not affect the creation of object files or the execution of the final application. Placing extremely large numbers of programs into a single workspace, however, can have a negative impact on AcuBench's performance, and can make managing the version control repository more complicated.

Before importing a project into the version control repository, consider the following:

- *Will each developer have a custom set of project modes (".pof" files) and one or more unique Tools/Options INI files, or will these files be created for and tracked with each project?*

  Creating and tracking customized project modes can help to create consistency in the building and testing process. Since ".ini" files created in the Tools/Options interface contain the defaults for screens, reports, and controls, maintaining a centralized ".ini" can help with consistency management. For more information about creating project modes, see **Chapter 7**. For more information about the ".ini" file, and the Tools/Options interface used to modify the file, see **Chapter 4, section 4.2, "The Tools/Options Dialog."**

- *Will each program be generated into a single ".cbl" file or several different COPY files (the AcuBench default)?*

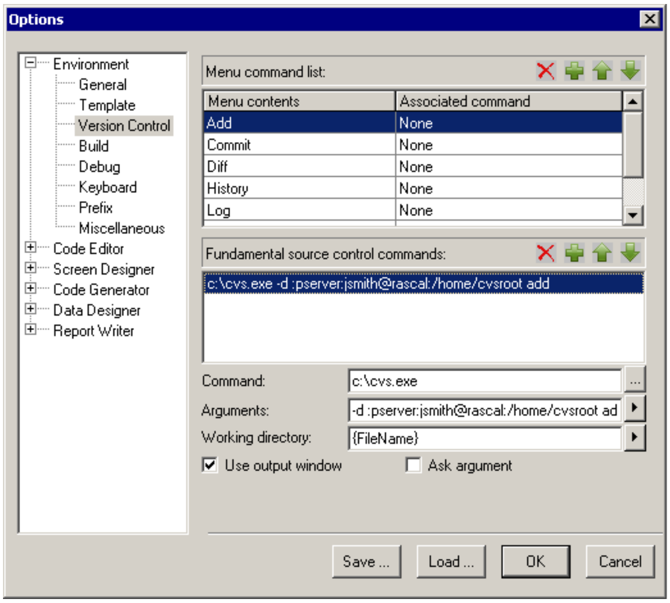
  Because all of the information generated into AcuBench COPY files is stored in the program structure file (".psf"), which must be tracked if you are using AcuBench code generation, it is more efficient to generate separate COPY files with version control. If all code is generated into the source (".cbl") file, then the same information is being tracked twice.

- *How will the project directory be structured?*

  Do you plan to use the default directories created by AcuBench, or are you using an existing directory structure? Keep in mind that some version control systems require all project files to reside a single directory structure (a main project directory plus subdirectories).

5.2 The Version Control Interface

You define, view, and maintain version control commands in the Tools/Options/Environment/Version Control window.



Menu contents	Associated command
Add	None
Commit	None
Diff	None
History	None
Log	None

c:\cvs.exe -d :pserver:jsmith@rascal:/home/cvsroot add

Command: c:\cvs.exe

Arguments: -d :pserver:jsmith@rascal:/home/cvsroot ad

Working directory: {FileName}

[✓] Use output window

[ ] Ask argument

The Options/Environment/Version Control Interface

Note: The Tools/Options/Environment/Version Control interface has substantially changed from Version 5.0. However, all Version Control commands defined in Version 5.0 projects are fully retained and can be modified and maintained in the revised interface.

5.2.1 Displaying the Version Control Window

You can display the version control window using the following steps:

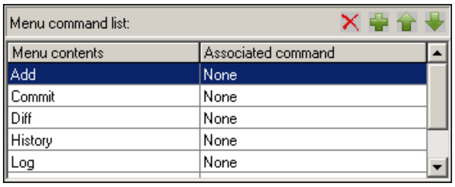
1. On the AcuBench menu bar, select **Tools/Options** to open the Options dialog.

2. Click the plus sign ("+") next to the Environment menu item to view its subitems.
3. Click **Version Control** to display the Version Control configuration window.

5.2.2 The Version Control Interface Fields

The following fields appear in the Version Control window:

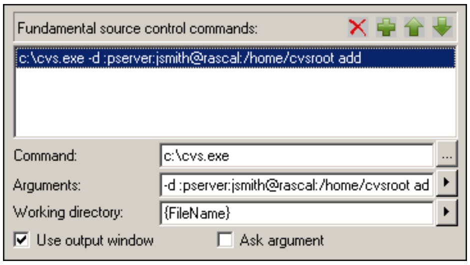
Menu command list


Menu command list:

Menu contents	Associated command
Add	None
Commit	None
Diff	None
History	None
Log	None

Displays the names of the commands that you have defined for your version control system in the "Menu contents" field. These are the menu items that will appear in the Version Control right-click menu. You can add, delete, and reorder commands using the four buttons at the top-right of the "Menu command" list. To modify a name, double-click the name and make your changes.

Fundamental source control commands


Fundamental source control commands:

```
c:\cvs.exe -d :pserver:jsmith@rascal:/home/cvsroot add
```

Command: c:\cvs.exe
Arguments: -d :pserver:jsmith@rascal:/home/cvsroot ad
Working directory: {FileName}
[✓] Use output window
[ ] Ask argument

Displays the command line(s) defined for the name selected in the "Menu command" list. Commands are defined in the "Command," "Argument," and "Working directory" fields and cannot be directly altered in this area. You can add or delete commands using the buttons at the top of the "Fundamental source control commands" list. To modify the definition of a command, select the command, then modify the values in the "Command," "Argument," and "Working directory" fields.

Command

In this field, enter the full path to and executable file name of your version control system software, or your batch file. You can search for the executable file by clicking the button next to the "Command" field.

Arguments

In this field, enter the appropriate arguments for the version control command. These arguments are sent to the version control system executable identified in the "Command" field. Arguments consist of commands recognized by your version control system. You may include any of a variety of variables defined by AcuBench. The values of these variables depend on the directory and file structure of your workspace. See the description included in **section 5.6**.

Working directory

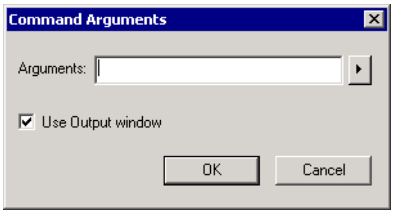
In this field, enter the path for the working directory from which the version control command must be executed. This directory must match the directory information set in your version control system. For example, if you have the working directory in your version control system set to C:\Checkout, then Working directory in AcuBench must also be set to C:\Checkout.

Use Output Window

When this option is enabled (checked), standard output from version control commands is redirected to the Version Control tab of the Output window.

Ask Argument

When this option is enabled (checked), a Command Arguments dialog box is displayed before AcuBench executes the version control command. The Command Arguments dialog allows the user to add arguments to the version control command.



This option can be uniquely set for every command listed in the "Fundamental source control commands" list. In addition, you can use any of the variables described in **section 5.6** to build your command-line argument.

5.3 Adding Commands to the Menu Command List

You can define a version control command as follows:

1. Click the **New** button at the top-right of the "Menu command" list and enter the name of the new command in the "Menu contents" field. This is the name that will appear on the Version Control pop-up menu.

2. With the new command still selected, click in the **Fundamental source control commands** area and click the **New** button. Define the command in the "Command," "Argument," and "Working directory" fields. Note that any command listed on the "Menu command" list can have multiple version control actions (commands) defined for it. To define another command, again click the **New** button in the "Fundamental source control commands" list and define the command in the "Command," "Argument," and "Working directory" fields. Commands are executed in the order they appear in the list. How to change the order is described in **section 5.4**.

5.4 Modifying the Command List

To delete or reorder your version control commands, first open the Tools/Options/Environment/Version Control dialog.

To delete a command from the “Menu command” list, select the command that you want to remove from the list and click the **Delete** button at the top-right. Note that all of the associated commands in the “Fundamental source control commands” list are also deleted.

Delete a command from the “Fundamental source control commands” list as follows:

1. Select the menu command with which the fundamental command is associated.
2. In the “Fundamental source control commands” list, select the command that you want to delete and click the **Delete** button at the top-right of the list. Note that the “Menu command” item is not removed.

To reorder the command names on the Version Control menu, in the “Menu command” list, select the command that you want to move up or down in the list and use the **Move Up** or **Move Down** arrow buttons to change the order. You can move only one command at a time.

5.5 Saving the Command List

To save your work in the current workspace, click **OK**.

To save your work in a separate “.ini” file, click **Save** at the bottom of the Options (Version Control) window and specify a name.

Caution: If you overwrite the “AcuBench80.ini” file, you are changing the default options that will be loaded by any new AcuBench workspaces that you create.

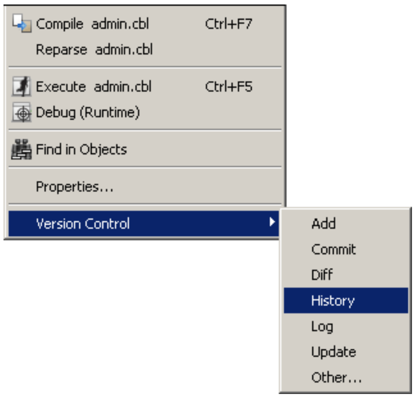
5.6 Command Variables

The following list describes the AcuBench-defined variables that you can include in the “Arguments” and “Working directory” fields. To insert a variable, click on the right-arrow button to the right of the entry field and select the desired variable from the list. AcuBench assigns values to these variables based on the state of your current workspace.

Variable	**Description**
File Path {FilePath}	The full path and name of the currently selected file or folder
File Directory {FileDir}	The full path to the directory of the currently selected file
File Name {FileName}	The name of the currently selected file
Current Directory {CurDir}	The path and name of the directory in which the AcuBench executable is located
Target Directory {TargetDir}	The path to the target project or folder. When source files are checked out, they are stored in the target directory until you check the files back in.
Workspace Path {WkspPath}	The full pathname and file name of the current workspace
Workspace Name {WkspName}	The name of the current workspace
Project Directory {ProjectDir}	The full path to the current project, excluding the name of the project file
Project Name {ProjectName}	The name of the current project
Folder Name {FolderName}	The name of the current folder.
Comment {Comment}	Include a comment defined by your version control system.

5.7 Issuing Version Control Commands

Once you have added commands to the Version Control interface, a new menu option, Version Control, is added to the pop-up menu that appears when you right-click a project node in any Workspace Window view, a program node in the Structure view, a file in the File view, or a data layout in the Data view.



The context for each command that you issue depends partially on which file is selected when you right-click and partially on the arguments that you have specified for the command. If, for example, you have specified that a "commit" command has {ProjectDir} specified, the entire project will be committed to the version control repository, regardless of which file you have selected. If, however, a "commit" command has {FileName} or {FileDir} specified, only the currently selected file is committed.

This means that you may define multiple menu options for a single version control command, each specifying a different context. As an alternative, you can specify the "Ask Arguments" option and manually specify a context each time you issue the command.

When you issue a version control command, AcuBench's default behavior is to echo the command to the Output window, followed by the message "Command sent, waiting for completion." Any messages sent back by the version control software are also displayed in the Output window. When the command is completed, AcuBench adds the message "Command completed—DONE" at the end of output listing.

The following example provides an idea of what appears in the Output window when you issue a version control command. Here, the cvs “log” command was used to return information about the AcuVet project. (Cvs is a free, open-source, third-party version control system.) The example includes only a portion of what was returned by the version control software.

```
=== cvs.exe -d :pserver:jramos@midas:/home/cvsroot log C:\AcuVet ... ===
Command sent, waiting for completion...
=============================================================================
cvs server: Logging C:/AcuVet
RCS file: /home/cvsroot/acuvet/AcuVet.pjt,v
Working file: C:/AcuVet/AcuVet.pjt
...
----------------------------
revision 1.2
date: 2006/07/27 21:56:42;  author: jramos;  state: Exp;  lines: +1 -1
Fixed the problems with the individual pet treatment history report.
Both the complete report and the individual report are part of the
"graph-rpt" program.  Both are accessed from the "Treatment History"
interface.  Use the button with the printer icon to print the
complete report and the button with the "T" icon to print the
individual report.
----------------------------
revision 1.1
date: 2006/07/20 19:38:52;  author: bsmith;  state: Exp;
branches:  1.1.1;
Initial revision
----------------------------
revision 1.1.1.1
date: 2006/07/20 19:38:52;  author: bsmith;  state: Exp;  lines: +0 -0
Creating a new testing project for AcuVet.
=============================================================================
...Command completed - DONE.
```

If the Output window displays an error message, rather than the expected result, it’s a good idea to double-check the context in which you are issuing the command. Some commands, for example, might work only when issued from the project directory or the root directory.

6 Working with Projects

Key Topics

AcuBench Project Management 6-2
Working at the Workspace Level 6-3
Working with Projects 6-7

6.1 AcuBench Project Management

Project development with AcuBench is organized into workspace, project, and program units.

An AcuBench *workspace* is a container for one or more projects. Physically, it consists of a dedicated file (*workspace_name*.pjt) that is created when the first project is created in or added to a new workspace. The workspace file records and maintains each member project's configuration and build options (the values of items set in Project/Settings), as well as other project-specific information and workspace configuration settings. The workspace concept is discussed in more detail in **section 3.2.1**.

An AcuBench *project* is a collection of all the files needed to build, run, and debug one or more programs, including all source files, COPY files, screen definition files, resource files (such as bitmaps and audio files), listing files, and object files. Each project has a set of dedicated folders for organizing project files. Each project's configuration and build options are recorded in the workspace file.

When a new project is established, AcuBench creates a home directory and a set of sub-directories for the project. You can specify which sub-directories are created, as well as sub-directory names, in the Tools/Options interface, under Environment/Prefix. Files can be added to the project with the Add/Remove Files function. Project files can reside directly in the project sub-directories or elsewhere on the file system, including on a network drive. AcuBench maintains logical links to project files.

Keep in mind that you control what's included in the project, which may be expanded to include ancillary items, or restricted to only those elements that are essential to the construction of the final product.

Support for the concurrent development of two or more projects is provided via the workspace structure. Projects that you want to work on concurrently must be members of the same workspace. To create multiple projects in the same workspace, see **section 6.3.1**.

6.2 Working at the Workspace Level

Having the workspace layer in AcuBench provides a means by which you can work with multiple projects simultaneously. This includes the capability to generate code for every program in a workspace, regardless of how many projects are in the workspace. It also provides a means for building all of the programs in a workspace.

It is worth noting that although there are no set limits on the number of programs that can reside in a single workspace, the code generation process is both memory-intensive and CPU-intensive. Also, a number of the handy automation features that AcuBench provides are likewise resource-intensive, so it's best to keep your workspace down to a reasonable size.

6.2.1 Creating a Workspace

Workspaces are created indirectly. To create a workspace you must create a project. The following conditions determine when a new workspace is created:

1. When you create a new project, if no workspace is open, AcuBench automatically creates a new workspace, giving it the same name as the new project.
2. If when you create a new project, a workspace is already open, you have the option of adding the new project to the current workspace or creating the project in a new workspace.

When a new workspace is created, it is automatically assigned the same name as the project that creates it. To avoid confusion, you may want to rename the project from which the workspace takes its name by right-clicking the project icon in any view of the Workspace window and selecting **Properties**. In the Project Properties interface, use the "Project name" entry-field to specify a new name for the project, then click **OK**.

To create a new project, see **section 6.3.1**.

6.2.2 Opening a Workspace

To open an existing workspace, select **Open Workspace** from the File menu. If you recently opened the workspace, you may also be able to open the workspace by clicking the workspace name in the File menu's Recent Workspaces list.

Note: You do not open individual projects. Every project is a member of a workspace, and it is the workspace that is opened, closed, and saved.

6.2.3 Saving a Workspace

To save all changes made to the current workspace, select **Save Workspace** from the File menu.

Note: When you select **Save** from the File menu, rather than Save Workspace, only the document that has focus is saved.

6.2.4 Building a Workspace

Rather than compiling your source programs one at a time, you can instruct AcuBench to compile all source files in a workspace (including their dependent COPY files) that have been modified since the last build. To do this, you can select **Build Workspace** from the Build menu or click the **Build** button on the Project toolbar.

The Build Button

To recompile all programs in the project, regardless of whether they have modified since the last build, use the **Rebuild Workspace** command, discussed in section 6.2.5.

By default, each program in the project is compiled according to the compile option settings of the selected project mode. To override the project-level compile settings, right-click the source file in the File view and select **Program Compile Options**.

During the build, the Output window displays the file name, compiler command line, and either a success ("completed") message or list of errors for each file as it is compiled.

If your workspace contains both local and remote objects, you must execute the Build or Rebuild Workspace command twice, once for local projects and once for the remote projects (after selecting the Build/Use Thin Client command). Note that for remote objects, the Build command causes the workbench to check the timestamp that the compiler places in the object file, not the UNIX file creation timestamp.

You can specify some attributes of the build and rebuild process through settings in the Tools/Options/Environment/Build dialog. In that dialog, you can specify such actions as whether the workbench always saves all files or generates code prior to starting compilation.

6.2.5 Rebuilding a Workspace

To instruct AcuBench to unconditionally compile all source files in the workspace, regardless of modification date, use the **Rebuild Workspace** command. Either select **Rebuild Workspace** from the Build menu or click the **Rebuild** button on the Project toolbar.

The Rebuild Button

In all other behaviors, the Rebuild command operates exactly like the Build command. See **section 6.2.4** for more details.

Note that when you use the command-line interface to build a workspace, the behavior described here is always used, regardless of whether you use the "build" or "rebuild" command.

6.2.6 Regenerating a Workspace

Use the Build/Regenerate Workspace command to cause all the code-generating elements of a workspace to be regenerated, including every program and screen, Working-Storage and Linkage Section items, all Event Editor code, and all code derived from each data set.

To streamline the process of regenerating and rebuilding a workspace, you can have AcuBench automatically regenerate the workspace each time that you issue a Build Workspace or Rebuild Workspace command. To do this, open the Tools/Options interface, expand the Environment tree, and select **Build**. Under "When building or reparsing the workspace," mark the box next to **Always generate modified program (.psf) files before compiling source files**.

If you use the AcuBench command-line interface to build a workspace, the workspace is automatically regenerated before source programs are compiled.

You can stop code generation by pressing **Enter** or **Esc**.

6.2.7 Stopping a Build

You can stop a build or rebuild in progress. To stop a build, click the **Stop Build** button on the Project toolbar. The Output window displays the names of all files compiled, as well as any error messages generated during the build up to the stopping point.

6.2.8 Closing a Workspace

To close a workspace, select **Close Workspace** from the File menu. If you made changes to components of the workspace that have not yet been saved, a dialog is displayed that asks whether you want to save the changes.

6.3 Working with Projects

Although you can create an empty project, projects are generally made up of programs and their COPY files and resources. Projects also contain data definitions and data handling code that can be used by any or all of the programs in that project. In addition, each project includes at least one set of compiler and runtime options.

By default, AcuBench creates a project directory and a series of subdirectories for each project. New programs created within the project are placed in the project directory, and generated COPY files are added to the appropriate subdirectories. The project grouping, however, is a virtual construct. Programs and files that appear in AcuBench's Workspace window do not need to reside within the default directory structure. In other words, it is possible to add existing programs to a project without moving those programs and their COPY files to the default project directory.

6.3.1 Creating a Project

When you create a project, AcuBench creates the infrastructure for the project, including a set of folders (working directories). Optionally, you can create a project that includes a templated source file, a templated screen, and a program structure file (".psf"). You can also create a project with object files directed to a remote server. For more about projects, see **section 3.2.2**.

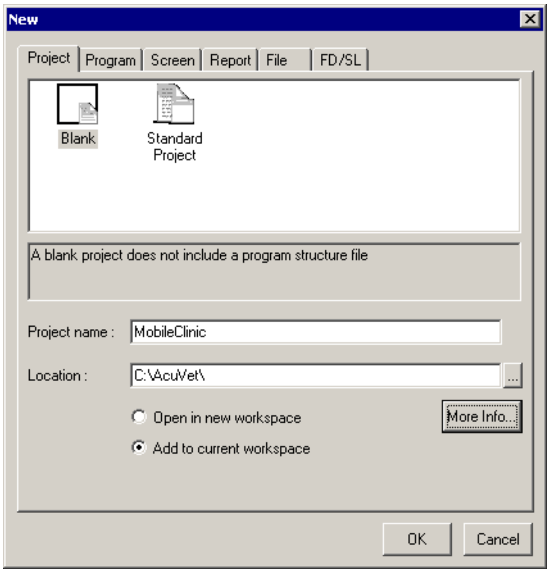
Once a project is created, you can change the name of the project, but you cannot change the name of the project's working directory or the names of any of the sub-directories.

Before you create a project, you need to decide whether to add the new project to the current workspace (if a workspace is open) or to create it in a new workspace. If you want to work on the new project concurrently with another project, open the workspace that contains that project before creating the new project.

Create a new project using the following procedures:

1. Select **New** from the File menu and click on the **Project** tab.

This opens the New dialog window with the Project tab selected.



A blank project does not include a program structure file

Project name : MobileClinic

Location : C:\AcuVet\

( ) Open in new workspace

(●) Add to current workspace

2. Select a project template.

   The Standard template creates a project and a program. The program, in addition to the program structure file, gets a standard graphical screen.

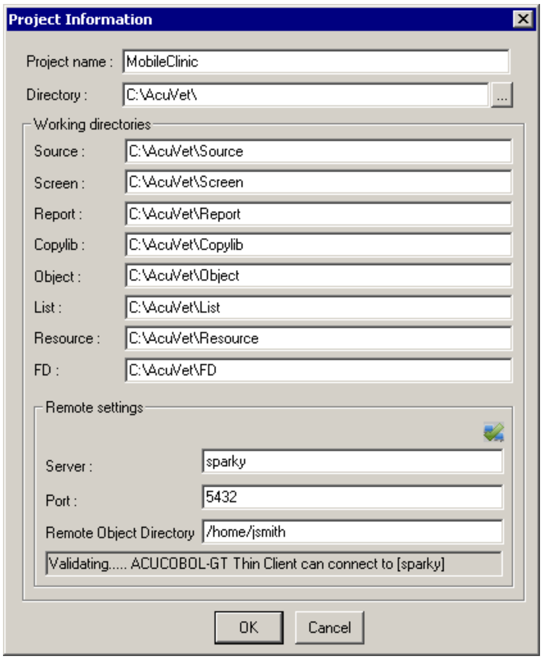
   The Blank template creates only an empty project structure, to which you can add programs and files.

3. Enter the name of the project in the **Project name** field. By default, the project directory is given the same name as the project, as shown in the **Location** field.

   Note that you can change the default path, and that multiple projects can reside in the same directory. Also note that the only files that must reside in this project directory are the workspace project file (".pjt") and the workspace information file (".wif").

4. Select one of the two radio buttons near the bottom of the Project tab to either include the project in the current workspace (this option is disabled if no workspace is open), or to create the project in a new workspace.

5. If you want to change the name of any of the standard project directories, click the **More Info** button and make the desired changes in the Project Information dialog.



Project Information

Project name : MobileClinic

Directory : C:\AcuVet\

Working directories

Source : C:\AcuVet\Source

Screen : C:\AcuVet\Screen

Report : C:\AcuVet\Report

Copylib : C:\AcuVet\Copylib

Object : C:\AcuVet\Object

List : C:\AcuVet\List

Resource : C:\AcuVet\Resource

FD : C:\AcuVet\FD

Remote settings

Server : sparky

Port : 5432

Remote Object Directory /home/jsmith

Validating..... ACUCOBOL-GT Thin Client can connect to [sparky]

This dialog is populated with information based on the project name that you have entered and the default project directory structure defined in the Tools/Options/Environment/Prefix interface. Information that you change in the Project Information dialog applies only to the individual project. To make changes to the default directory structure established for any new project that you create, use the Tools/Options interface, as described in section 4.3.7.

In the top portion of the Project Information dialog, you can change the name of the project and the project directory. In the middle portion of the dialog, you can change the default location for each of the file types (source, screen, report, and so on) associated with an AcuBench project.

6. If you want your project's object files to reside on a remote server, use the remote settings area of the Project Information dialog to enter the server name, port number, and remote object directory for your project.

You can use the **Validate** button (located at the upper, right corner of the remote settings area) to determine whether AcuBench can make the remote connection. If the remote server is up and AcuConnect is running, the validation should succeed. If it does not succeed, you will receive one of the following error messages:

- "ACUCOBOL-GT cannot connect to *servername*." This may indicate, among other things, that the server name that you entered is invalid, that you do not have a network connection, or that the server is not listening on the specified port.
- "*remotedirectoryname* is invalid, or does not exist." This may indicate that the specified directory does not exist, that you are using incorrect directory syntax, or that you do not have permissions to access the directory.

7. Click **OK** to create the project.

To add files and resources to your project, click the **Add/Remove Files** button on the Project toolbar (see **section 6.3.4**).

Note: By default, you do not open, close, and save individual projects. Every project is a member of a workspace, and it is the workspace that is opened, closed, and saved.

It is possible, however, to save information about a single project in a project file (".pjf"). Just right-click the project icon and select **Save Project - *projectname***. You can then move or copy the project to a new workspace using the File/Open Project command).

6.3.2 Project Properties

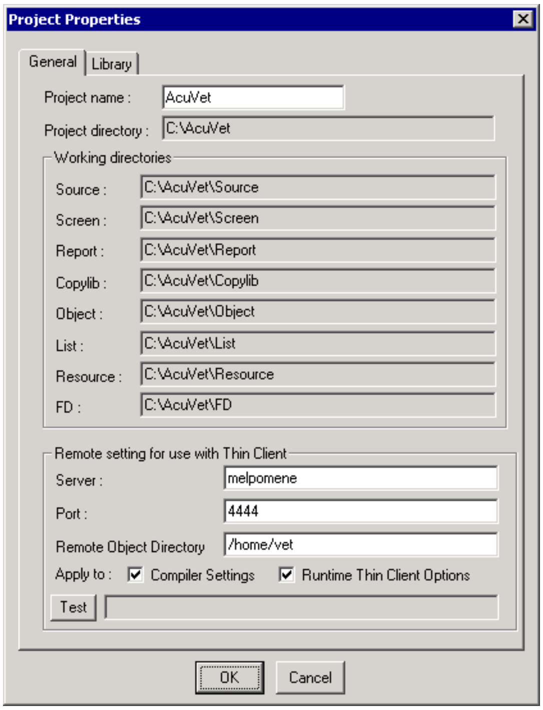
A project's properties include its name, its location, and the names of its local and remote working directories. These properties are established when the project is created, and some of them, including the project's location, cannot be changed.

You can view the project's properties and change the project's name in the Project Properties dialog. You can also establish settings to use with the AcuConnect thin client at the bottom of the dialog's General tab. In addition to specifying a server, port, and remote object directory, you can also determine whether the thin client settings will apply at compile time, run time, or both.

The Project Properties dialog also includes a Library tab that is used to specify the primary module and module type when you create an object library. For information about creating object libraries, see **Chapter 7, section 7.6, "Library Settings,"** and **Chapter 19, section 19.2, "The Object File Utility."**

Note: For information about setting project compile, runtime, and environment options, see **Chapter 7, "Chapter 7: Project Settings."**

To display project properties, right-click the desired project node in the Workspace window and select **Properties** from the pop-up menu.


Project Properties

General | Library

Project name : AcuVet
Project directory : C:\AcuVet

Working directories
Source : C:\AcuVet\Source
Screen : C:\AcuVet\Screen
Report : C:\AcuVet\Report
Copylib : C:\AcuVet\Copylib
Object : C:\AcuVet\Object
List : C:\AcuVet\List
Resource : C:\AcuVet\Resource
FD : C:\AcuVet\FD

Remote setting for use with Thin Client
Server : melpomene
Port : 4444
Remote Object Directory /home/vet
Apply to : [✓] Compiler Settings [✓] Runtime Thin Client Options

You can rename a project as follows:

1. Open the Project Properties dialog by right-clicking the desired project node and selecting **Properties** from the pop-up menu.

2. Enter the new project name in the Project Name field and click **OK**.

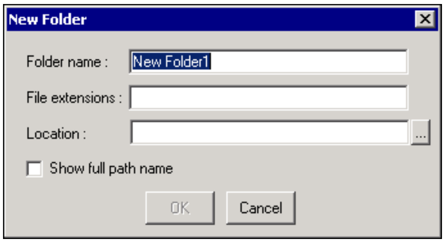
Although the new project name appears in the Workspace window's views, the name of the workspace project file ("*workspace_name*.pjt") is not changed. If you have created an individual project file ("*project_name*.pjf") for the project, the existing file is not affected by the change. The next time that you issue a **Save Project** command, you are given the option to change the name of the ".pjf" file.

6.3.3 Adding Folders to a Project

You can add folders to the set of default project folders. Such folders can be used for any purpose that you choose.

Add a new folder using the following steps:

1. In the workspace File view, right-click on the project node that you want to add a folder to and select **New Folder** from the pop-up menu.



2. Specify a name for the new folder in the "Folder name" field.

3. Enter the file extensions that you want to associate with the folder in the "File extensions" field. For example, if you want a folder specifically for bitmap graphics and icons, enter "*.bmp; *.ico" in this field. You must enter a value. You can use "*.*" to indicate all file extensions.

4. Specify a location where files associated with the folder will reside. Click the **Browse** button ("...") to the right of the field to navigate the file system to select a location. If the specified location does not exist, it is created automatically. For example, to create a new physical folder for documentation files in the same location as your default project folders, your location specification might look like:

   ```
   c:\development\project1\documentation
   ```

A new physical folder named “documentation” is created when you click OK to complete the dialog.

Note: Note that only new files created in the workbench or files that you physically move to the folder are actually located there. When you add an existing file to the folder with the Add/Remove File function, AcuBench simply creates a logical link to the file.

---

5. Enable the **Show full path name** check box to cause the full path name of files in this folder to display in the File view.
6. Click **OK** to create the folder.

6.3.4 Adding Files to a Project

Although you open, close, and save the workspace, you add files to and remove files from individual projects. The Add/Remove Files function is the vehicle for adding files to and removing files from a project.

Caution: Because you can have multiple projects in the same workspace, there is the potential for inadvertently adding files to or removing files from the wrong project. Select the target project in the Workspace window before initiating the Add/Remove Files function. To select a project in the Workspace window, click on the desired project node, or one of its subnodes, in any workspace view. Alternatively, you can select the target project in the “Insert into” combo box at the bottom of the Add/Remove Files dialog.

The Add/Remove Files dialog has several tabs, one for each category of file type (by default: Source, Screen, Report, Copylib, Object, List, Resource, and FD).

This section gives an overview of the steps involved in adding any file to a project. Specific information about adding source files to a project is provided in **Chapter 9, section 9.3.2, “Adding an Existing Source File.”**

Note: To move a file from one project or folder to another project or folder, simply select it in the File view and drag it to the target folder. A similar cut-copy-and-paste function is not supported.

You can add a file to a project as follows:

1. Right-click a folder in the File view and select **Add/Remove Files**. This opens the Add/Remove Files dialog with the tab appropriate to the selected folder in the foreground.

   You can also select the desired project in the Workspace window and click on the **Add/Remove Files** icon on the Project toolbar, or select the **Add/Remove Files** option on the Project menu.

2. Navigate to the folder containing the file to be added. A list of files that match the suffixes in the “Files of type” field is displayed in the “Files list” (top left).

   If the wrong file types are listed, use the “Files of type” drop-down box to select a different set of file extensions. The available file extensions reflect the file types associated with the project in the properties for each File view folder.

3. Select (click on) the file to be added in the “Files list”. The file must be highlighted.

   To select multiple consecutive files, hold down the **Shift** key and click the files. To select multiple non-consecutive files, hold down the **Ctrl** key as you click.

4. To add the selected file(s), click **Add**. To add all of the files in the specified directory, click **Add all**. The added files now appear in the “Files in project” list box.

5. When you have finished adding files, click **OK** to close the Add/Remove Files dialog.

6.3.5 Removing Files From a Project

You can remove files from a project with the Add/Remove Files dialog. Note that although the files are removed from the project, they are *not* deleted from the disk.

Caution: Because you can have multiple projects in the same workspace, there is the potential for inadvertently removing files from the wrong project. Always select the target project in the Workspace window before initiating the Add/Remove Files function. To select the desired project, click on the desired project node, or one of its subnodes, in any workspace view.

The Add/Remove Files dialog has several tabs, one for each category of file type (by default: Source, Screen, Report, Copylib, Object, List, Resource, and FD).

You can remove files from a project as follows:

1. Select the desired project in the Workspace window and click on the **Add/Remove Files** icon on the Project toolbar, or select the **Add/Remove Files** option on the Project menu. The Add/Remove Files dialog is displayed.
2. Click on the tab that corresponds to the type of file to be removed.
3. In the “Files in project” list, select the files to be removed and click the **Remove** button.
4. When you are done, click **OK** to accept the changes and close the dialog. If you press Cancel, your changes are not recorded.

Note: You can quickly remove all of the files of a selected type by clicking the **Remove All** button.

To empty a folder, right-click on the folder that you want to remove all files from in the File tab of the Workspace window. Select **Empty** from the pop-up menu.

6.3.6 Moving Components Among Projects and Folders

From within the Workspace window, you can easily move a program, screen, file, or data layout from one project or folder to another in the same workspace.

Moving a Program, Screen, Report, or Data Layout to Another Project

In the Workspace window, locate the project component to be moved and drag-and-drop it on the target project node. The component is placed in the appropriate area. If a component of the same name is already present in the target project, the item is not moved and a message is displayed.

Moving a File to Another Folder or Project

In the File view, locate the file to be moved and drag-and-drop it on the destination folder. The file is removed from the initial folder and placed in the target folder.

Note: The file is not physically moved; rather the logical link is simply changed. Also note that the workbench does not restrict the type of file that can be moved into another folder, nor does it generate a warning when the file type differs from the type that the target folder is designed to host. As a result, it is easy to move, intentionally or unintentionally, a file of one type into a folder meant for another type (for example, a source file into the Object folder).

6.3.7 Deleting a Project

To delete a project from a workspace, right-click on the project node in the Workspace window and select **Delete** from the pop-up menu. You can also select the project node and press the Delete key.

7 Project Settings

Key Topics

Introduction 7-2
Modes 7-3
Compiler and Runtime Settings 7-6
Working with Runtime Configuration Files 7-8
Environment Settings 7-11
Library Settings 7-15

7.1 Introduction

AcuBench provides an interface for specifying compiler, runtime, library, and environment settings at the project level. You can create multiple sets of settings and switch between them quickly and easily to change how programs in your projects are built and executed.

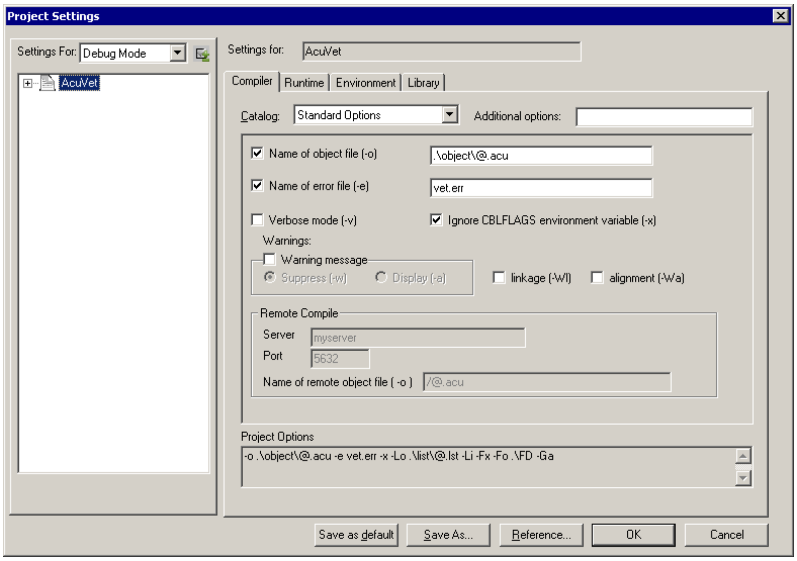
- You can define distinct project option settings for each AcuBench project.
- You can have individual compile options for any COBOL source file.
- You can define sets of settings (*modes*) that enable you to quickly shift to settings that meet special needs, such as settings that support debugging work, or settings that build the program for 64-bit systems.
- You can save modes that you create and load the saved modes into other projects.

You specify these settings in the **Project Settings** interface, which can be accessed in any of the following ways. First select a project icon in any Workspace window view, then:

- Select **Settings** from the Project menu.
- Click the **Settings** button on the Project toolbar.
- Right-click the project icon and select **Settings** from the pop-up menu.

The Project Settings interface opens with the Compiler tab selected and the three other tabs—Runtime, Environment, and Library—in the background. All projects residing in the current workspace are listed in a tree view on the left side of the screen, and a field above the tabs indicates which project is currently selected. It is important to note which project you are working in, because settings created for one project in a workspace are not automatically transferred to other projects in the same workspace. All programs in a project, however, use the settings established at the project level unless you specifically define special settings for an individual program.

The Project Settings interface looks like this:



Settings For: Debug Mode

AcuVet

Settings for: AcuVet

Catalog: Standard Options

Additional options:

✓ Name of object file (-o) .\object\@.acu

✓ Name of error file (-e) vet.err

☐ Verbose mode (-v)

✓ Ignore CBLFLAGS environment variable (-x)

Warnings:

☐ Warning message

● Suppress (-w) ○ Display (-a)

☐ linkage (-Wl) ☐ alignment (-Wa)

Remote Compile

Server myserver

Port 5632

Name of remote object file (-o) /@.acu

Project Options

```
-o .\object\@.acu -e vet.err -x -Lo .\list\@.lst -Li -Fx -Fo .\FD -Ga
```

7.2 Modes

In AcuBench, the term *mode* is used to refer to a set of project settings. Each project must have at least one mode and may have many modes. The ability to have multiple modes makes it easy to quickly switch from one set of compiler switches, for example, to another. This can provide a straightforward way for you to compare the effects of different compiler, runtime, and environment settings on program performance.

By default, AcuBench provides a Release mode, containing very simple compiler and runtime command lines and basic environment settings, and a Debug mode, which adds the "-Ga" compiler flag to include all available debugging information in the compiled object.

When you make changes to an existing mode or add a new mode in the Project Settings interface, the changes are saved in the workspace project file (".pjt") for the affected project. If you create an individual project file (".pjf"), these settings are carried over to that file.

You can also store each mode in a project options file (".pof"). Project options files and methods for creating them are discussed in **section 7.2.4**.

7.2.1 Working with Modes

In the Project Settings dialog, above the tree view listing all of the projects in the current workspace, a "Settings For" drop-list displays the current mode. If you have more than one mode defined for a project, or more than one project defined in a workspace, it is essential to make sure that you are working in the correct mode for the correct project.

Settings made to a mode in one project do not affect any other project, even if both projects have a mode with the same name. Likewise, changes made to one mode in a project do not affect any other mode in the same project.

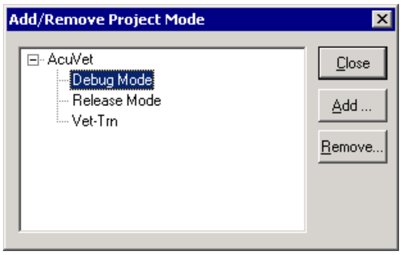
Consider the following. You have a workspace containing two projects: MainPrj and Alternate. You have defined a custom Production mode for MainPrj and want to change the FILE_PREFIX setting for that mode. If you enter the Project Settings dialog and select Production mode, but fail to notice that Alternate is selected as a project, when it comes time to execute a program in MainPrj, you will either receive a file error or access the wrong set of data files, because the FILE_PREFIX setting will be incorrect.

7.2.2 Adding a Mode

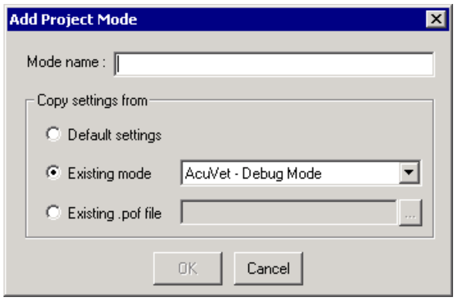
To add a mode to your project, do the following:

1. In the Project Settings window, select a project in the tree view, then click the **Add/Remove Mode** button (to the right of the "Settings For" drop-list, above the tree view list).

   This opens the Add/Remove Project Mode dialog.



2. Click **Add**. This opens the Add Project Mode dialog.



3. Enter a name for the new mode, then select one of the radio buttons in the “Copy settings from” area to determine which existing mode, if any, to use as the basis for the new mode.

   - To create a new mode based on the default settings established in the “default.pof” file, select **Default settings**.
   - To create a new mode based on another mode defined in this project, select **Existing mode**, then select a mode from the drop-down list.
   - To create a new mode based on an existing “.pof” file, select **Existing .pof file**, then click the browse (...) button to navigate to the location of the file on disk.

   POF files are discussed in **section 7.2.4**.

4. Click **OK**, then click **Close**. This returns you to the Project Settings interface to begin entering settings for your new mode.

7.2.3 Removing a Mode

To remove a mode from a project:

1. In the Project Settings window, select a project in the tree view, then click the **Add/Remove Mode** button (to the right of the “Settings For” drop-list, above the tree view list).
2. Select a mode in the Add/Remove Project Mode list, then click **Remove**.

3. An AcuBench message asks you to confirm the removal of the mode. Click **Yes**.

4. Click **Close** to return to the Project Settings interface.

7.2.4 Saving a Mode as a POF

A project options file (".pof") is a text file used to store project modes. Project modes currently in use in a project are stored in the workspace project file (".pjt") and updated every time you click **OK** to save your settings and exit the Project Settings interface.

Note that project options files are static. In other words, once you write options to a POF, those options are not automatically updated when you change the corresponding mode in the project from which the POF was created. To update the settings in a POF, you must explicitly save the changes to a file.

There are two ways to save a mode as a POF:

- Click **Save As** to specify a file name and location for the new settings file. This file can later be used to add a new mode to any AcuBench project.

- Click **Save as default** at the bottom of the Project Settings window to write your changes to a file called "default.pof" in the AcuBench installation directory. Every new project that you create will take its settings from this file.

  The mode saved in "default.pof" is always called Release, and it is always used to create a second default mode, called Debug, by adding the "-Ga" compiler flag to the existing compiler command line.

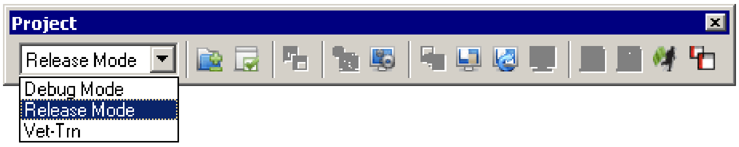
7.2.5 Reusing a POF

As discussed in **section 7.2.2**, when you add a new mode to a project, you have the option to base that mode on an existing POF file, specified through the Add New Project Mode interface. This is one of two ways to make use of an existing POF.

When you are working in the Project Settings interface, you can click **Reference** to replace settings used in the current mode with settings from a POF. Although this changes the compiler, runtime, environment, and library settings for the mode that you have selected, the name of the mode stays the same.

7.2.6 Switching Between Modes

Once you have defined modes for your project, you can easily switch between them as you work in AcuBench. The Project toolbar lists all of the modes defined for projects in your workspace in a drop-down list.



To change the current mode, first select a project in any Workspace window view, then select a mode from the drop-down box. The next time that you compile or execute programs in the project, the new settings are used.

7.3 Compiler and Runtime Settings

Settings established on the Compiler tab of the Project Settings dialog are, by default, applied to each program in the specified project at compile time. You can override these project-level settings at the individual program level by selecting a *program* in the Project Settings tree view, rather than a project, and de-selecting the **Follow project default options** check box.

Settings defined on the Runtime tab of the Project Settings dialog are applied to each program in the specified project when the program is executed.

Both of these two tabs are constructed in a similar manner. A "Catalog" drop-down list at the top of the tab lists categories of options that you can modify. Each compiler catalog corresponds to a section in Chapter 2 of the *ACUCOBOL-GT User's Guide*. Each of the options in the four runtime catalogs is also documented in that chapter, but because there are many fewer runtime options than compiler options, the documentation does not separate them into individual sections.

When you select a catalog, the compiler and runtime command-line options (also called switches or flags) that correspond to that catalog are displayed on the tab. For example, the "--javaclass", "--javamain", "--netdll", and "--netexe" compiler options can be found on the Interoperability catalog of the Compiler tab, and the "--wait" and "--restart" runtime options can be found on the Thin Client catalog of the Runtime tab. In most cases, you can mark a check box or select a radio button to add a flag. Occasionally, an entry field or dialog may also be associated with an option.

As you make changes to your settings, the corresponding runtime or compiler command line appears in the Project Options field at the bottom of the screen. This field is disabled, preventing you from making changes to this command line directly.

An "Additional options" field in the top, right corner of the runtime and compiler tabs allows you the option of entering command-line flags directly. While it is preferable, in most instances, to use the various catalog options to change your settings, it is sometimes necessary to use the Additional options field to invoke a switch has not been added to the AcuBench interface.

Text that you enter in the Additional options field is added to the command line in the Project Options field at the bottom of the screen as you type. This makes it possible for you to verify that the command line you are entering is correct.

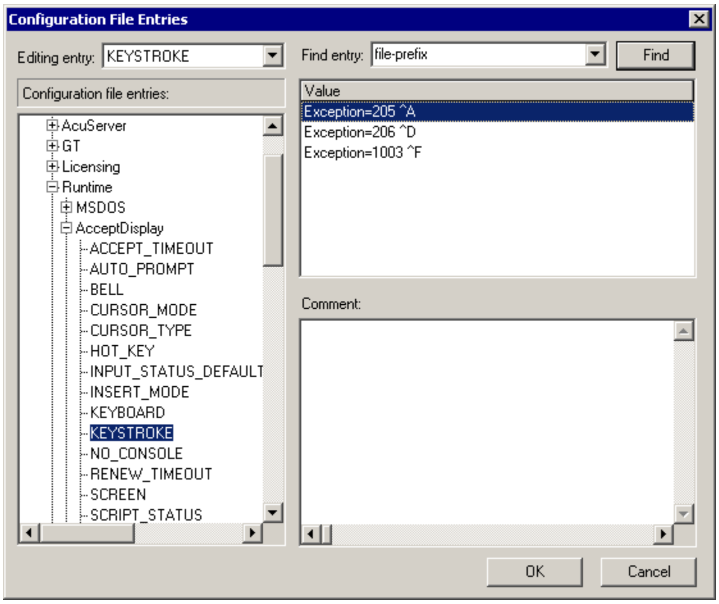
Note: The "-Po" precompiler option was discontinued with the Version 8.0 AcuSQL. The Version 8.0 AcuBench strips the "-Po" if it is found in the project when it is being opened. If you are using Version 8.0 AcuBench with a pre-Version 8.0 compiler, you will need to add "-Po" back manually in the "Additional options" field.

7.4 Working with Runtime Configuration Files

AcuBench includes an integrated Configuration File Editor (CFE) that allows you to open and edit any ACUCOBOL-GT runtime configuration file. With the CFE, you no longer need to access the host system's text editor to create new configuration files or modify a configuration variable.

The CFE provides a list of most of the available ACUCOBOL-GT configuration variables, along with an interface that allows you to add your own user-defined variables. Both types of variables can be listed and modified within the CFE in the same manner.

The CFE is accessed through the Runtime tab of the Project Settings interface. When the Common Options catalog is selected, the "Alternate configuration file name (-c)" option allows you to specify a configuration file that will be used when you execute programs in your project. Enter a path and file name here, then click the **Edit** button to launch the CFE.


Configuration File Entries

Editing entry: KEYSTROKE

Find entry: file-prefix

Configuration file entries:

```
⊞ AcuServer
⊞ GT
⊞ Licensing
⊟ Runtime
  ⊞ MSDOS
  ⊟ AcceptDisplay
    ACCEPT_TIMEOUT
    AUTO_PROMPT
    BELL
    CURSOR_MODE
    CURSOR_TYPE
    HOT_KEY
    INPUT_STATUS_DEFAULT
    INSERT_MODE
    KEYBOARD
    KEYSTROKE
    NO_CONSOLE
    RENEW_TIMEOUT
    SCREEN
    SCRIPT_STATUS
```

Value
Exception=205 ^A
Exception=206 ^D
Exception=1003 ^F

Comment:

When you enter a path to a configuration file that already exists, the CFE opens and parses the selected file, even if the file was not created in AcuBench.

Currently the CFE has one notable limitation. Although it can be used to add or remove a configuration variable, it cannot be used to "comment out" or "uncomment" a variable (place or remove a "#" in front of a variable).

7.4.1 Editing Configuration Files

When you edit a configuration file, the "Configuration file entries" list on the right side of the screen shows ACUCOBOL-GT configuration variables, organized by category. These categories include:

- **FileSystem**, which includes operating system environment variables such as those relating to the default host, file names, or to the Vision file system.
- **GT**, which lists ACUCOBOL-GT compiler-related configuration variables.
- **Licensing**, which holds license management configuration variables.
- **Runtime**, which includes runtime configuration variables.
- **User Defined**, which lists other variables defined in the configuration file.

Variables to which a value has been assigned appear in the "Editing entry" drop-down list in the top, left portion of the screen. When you select an option from this list, the CFE highlights the variable name in the tree view along the left side of the interface and shows the value assigned to the variable in the Value list in the top, right portion of the screen.

To add or change the value of a configuration variable, do the following:

1. Locate the variable in the interface using one of the following methods:
    - If a value has already been assigned to the variable, select the variable from the **Editing entry** drop-down list.
    - Navigate the tree view on the left side of the screen to locate the variable you want to edit.

- Enter the configuration variable name or some portion of it in the **Find entry** field, then click **Find** or press **Enter**.

  Note that this find function searches from the beginning of the word only. This means that the string "key" can be used to search for the KEYBOARD and KEYSTROKE variables, but not for the HOT_KEY variable.

2. Double-click in the **Value** area to change the variable's value. The behavior of the Value field varies depending on the value that you have selected.

   - If the variable takes a directory path (or multiple paths) as its value, a browse (...) button will appear in the field. When you click this button, a dialog opens, allowing you to navigate to the path. This can help to prevent typing and syntax errors.

   - With variables like KEYSTROKE, which can appear multiple times in a file, assigning unrelated values, multiple lines of entry are enabled in the Value column.

3. If you want to associated a comment with the selected variable in your configuration file, click in the **Comment** field.

   Remember that while the editor allows you to add comments about existing variables, it does not provide a method for commenting or uncommenting the variables themselves.

4. To remove a configuration variable from the configuration file, highlight the entry in the Value column, then press **Delete**. If you have previously assigned a comment to the variable, use the **Delete** or **Backspace** key to remove that, as well.

   Configuration variables that are not assigned a value are removed from the configuration file.

5. To save your changes, click **OK**.

   Your changes are written to the specified text file.

7.4.2 User-defined Variables

In addition to the standard set of ACUCOBOL-GT configuration variables, the CFE allows you to add your own variables. If you are using multiple file systems, for example, to access data, you may need to add *filename*_HOST entries for various data files that your programs access. You might also want to define your own variables to help set user access privileges or otherwise adapt the behavior of your programs in different environments.

To add a new configuration variable to the CFE:

1. In the "Configuration file entries" tree view, right-click on the **User Defined** category and select **New** from the menu.
2. Type the variable name, then press **Enter**.
3. Click in the **Value** column, then enter a value for your variable.

   If you have accidentally entered the name of a variable that has already been defined in this file, the existing variable value will appear in the Value list.
4. To add a comment to the variable definition in the configuration file, click in the **Comment** box and add text.
5. To delete a user-defined variable both from the configuration file and from the "Configuration file entries" tree, right-click the variable name and select **Delete**.
6. When you finish editing the file, click **OK** to save the entry and close the CFE.

   Once user-defined variables have been added, they are stored in the "User Defined" category in the "Configuration file entries" tree view. You can locate and edit them as you would pre-defined variables.

7.5 Environment Settings

The Environment tab of the Project Settings window is used to set the environment variables that apply to the selected project. You can view, add, modify, or delete environment variables in this interface.

Five environment variables are predefined by default. They are:

ACUPATH	defines the path to the ACUCOBOL-GT executable files (e.g., c:\Acucorp\Acucbl8xx\AcuGT\bin)
CODE_PREFIX	defines directories to search when a program name is specified without a full path. See Book 1, Chapter 2 of the ACUCOBOL-GT documentation set.
COMPILERNAME	defines the name of the compiler executable file (by default, “ccbl32.exe”)
COPYPATH	defines one or more paths that AcuBench searches to locate COBOL COPY files (see **section 7.5.2**)
RUNNAME	defines the name of the runtime executable file (by default, “wrun32.exe”)

Other variables are added when AcuBench verifies that it can connect to a specified remote server and port:

THINNAME	defines the name of the thin client executable (by default, “acuthin.exe”)
CLASSPATHDIR	defines the name of the path containing the AcuXUI Java classes
XUIJAR	defines the name of the AcuXUI JAR file

As with the RUNNAME executable, the executable defined by THINNAME and the directory defined by CLASSPATHDIR must be located in ACUPATH.

If you want to pass Java parameters as part of an AcuXUI Java command line, you can add a variable called XUIPARAMS to the project’s environment. For the value of XUIPARAMS, enter the Java parameters that you want to pass.

Caution: The value of variables defined in the Environment tab take precedence over the values of corresponding variables defined in the runtime configuration file. This is especially important to note if you have set CODE_PREFIX in your configuration file. For more information, see Chapters 1 and 2 in the *ACUCOBOL-GT User’s Guide* and Appendix H in *ACUCOBOL-GT Appendices*.

7.5.1 Working with Environment Variables

You can use the Environment tab of the Project Settings dialog to add, modify, and delete environment variables.

To add an environment variable for the current project:

1. Click the **New** button in the top, right corner of the Environment tab. This is the first of the three available buttons.

   The “Variable” and “Value” fields below the “Environment variables” list are cleared.

2. Enter a variable name in the “Variable” field.

3. Enter a value in the “Value” field.

4. Click the **Set** button. (**Set** is the third of the buttons in the top, right portion of the Environment tab.) The new environment variable is added to the list and the Variable and Value fields are cleared.

To change the value of an existing environment variable, select the variable in the “Environment variables” list, then make the necessary changes in the **Value** field. When you are finished, click **Set** to save your changes.

To remove an environment variable, select the variable from the “Environment Variables” list and click **Delete** (the second of the three buttons in the top, right portion of the Environment tab).

7.5.2 The COPYPATH Environment Variable

The value of the environment variable COPYPATH is used by AcuBench (and the ACUCOBOL-GT compiler) to search for COBOL COPY files. When AcuBench (or the compiler) needs to access a COPY file, it searches, from first to last, the directories specified in COPYPATH.

You can define COPYPATH to specify one or more relative or absolute directory paths. By default, COPYPATH is set to the relative paths of the project’s screen, COPY, resource, report, and FD folders (%screendir%, %copydir%, %resdir%, %reportdir%, and %layoutdir%). It also includes the absolute path to the sample directory that contains ACUCOBOL-GT “.def” COPY files (“C:\Acucorp\Acucbl8xx\AcuGT\sample\def”).

You can change the value of COPYPATH in one of two ways:

1. Use the Environment tab of the Project Settings dialog.
2. Add new paths as you add new source files to your project.

   The exact method by which new paths are added when you add source files to your project is determined by the settings that you have chosen in the Environment/Miscellaneous category of the Tools/Options dialog (discussed in **Chapter 4, section 4.3.8, "Miscellaneous Environment Options."**). If both "Automatically modify COPYPATH" options are selected:

   - When you add a new source file and select the **Automatically parse program for COPY files** option, COPYPATH is updated each time a new COPY file directory is identified.
   - When you use the **Copy Path** button in the Add/Remove Files dialog to indicate a new COPY file location, that path is added to the COPYPATH definition.

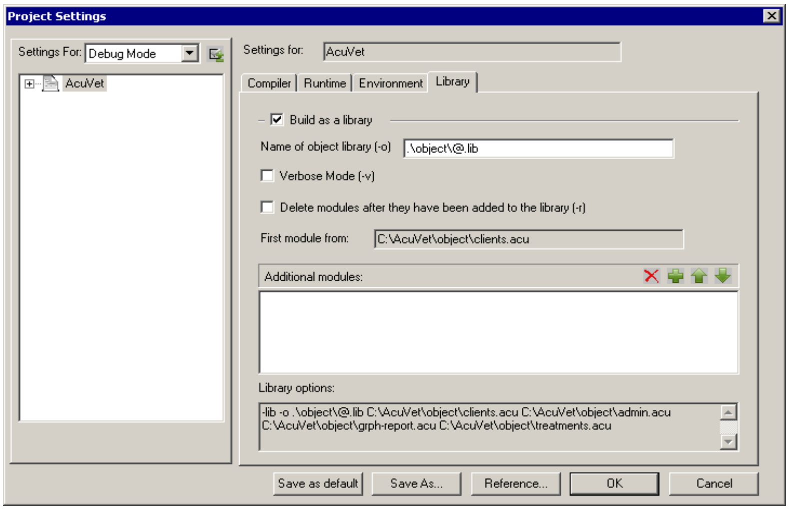
Special Notes on Changing COPYPATH

- Use caution when changing the value of COPYPATH. Changing the value of COPYPATH can potentially affect every source file in the project. For example, if you change the definition of COPYPATH to remove a path that holds a COPY file needed by the project, some files may no longer compile.
- If you add a path that holds a COPY file that has the same name as a COPY file in another directory that is also defined in COPYPATH, the COPY file used is the first COPY file found (the paths are searched from first to last).
- If you have a file or set of files that need a special COPY directory setting that you do not want to add to the definition of COPYPATH, you can specify that path on a file-by-file basis using the "Directories to be searched for COPY (-Sp)" option in the Source Options catalog of the Project Settings dialog. To specify a path in this way, right-click the target file and select **Program Compile Options** from the pop-up menu. The Project Settings dialog is displayed. Select the **Source Options** item from the Catalog drop-down list and check the **Directories to be searched for COPY** option. Specify a path (or paths) in the adjacent

entry box. Follow this procedure for each file that requires a special path. Setting this option overrides the COPYPATH environment variable for that file. The specified value is used every time that file is compiled and whenever the workbench needs to locate a COPY file for that file.

7.6 Library Settings

An object library is a file that contains a group of one or more compiled ACUCOBOL-GT programs and its associated resource files (bitmaps, WAV files, and so on). Ordinarily, if you wanted to create an object library, you would invoke **cblutil**, the object file utility (discussed in **Chapter 19** of this manual and Chapter 3 of the *ACUCOBOL-GT User's Guide*). If, however, you want to automatically create an object library every time you build a project, you can use the Library tab of the Project Settings window to define the modules of that library file.



Settings for: AcuVet

[✓] Build as a library

Name of object library (-o) .\object\@.lib

[ ] Verbose Mode (-v)

[ ] Delete modules after they have been added to the library (-r)

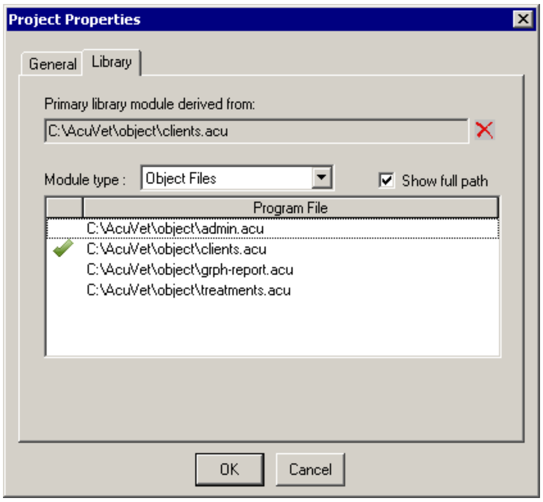
First module from: C:\AcuVet\object\clients.acu

Library options:

```
-lib -o .\object\@.lib C:\AcuVet\object\clients.acu C:\AcuVet\object\admin.acu
C:\AcuVet\object\grph-report.acu C:\AcuVet\object\treatments.acu
```

To specify that you intend to build an object library at compile time, mark the **Build as library** check box. When you do this, all of the object files in your project are marked for addition to the library. You can add additional object files or libraries (residing in other projects or directories) in the "Additional modules" list. When you build the project, AcuBench invokes **cblutil** using the command line shown in the "Library options" field at the bottom of the Library tab.

Although the Library tab provides a method for changing the order of additional modules added to the object library from outside the project, it does not provide a means for ordering the modules within the project, nor for declaring a primary module. The process of declaring a primary module for the object library is performed in the Project Properties interface:



Primary library module derived from:
C:\AcuVet\object\clients.acu

Module type : Object Files

☑ Show full path

	Program File
	C:\AcuVet\object\admin.acu
✓	C:\AcuVet\object\clients.acu
	C:\AcuVet\object\grph-report.acu
	C:\AcuVet\object\treatments.acu

To specify a primary module:

1. Right-click the project icon in the Workspace window and select **Properties**.

2. In the Project Properties interface, select the **Library** tab.

   The Library tab lists all of the object files associated with the current project.

3. In the Program File list, double-click an item to select it as the primary module. The item that you select is indicated by a check mark and added to the "Primary library module derived from" field.

   To change the item selected as the primary module, double-click another item in the Program File list.

4. When you are finished, click **OK** to save your changes.

The changes that you make in this Project Properties interface are used to update the **cblutil** command line in the Project Settings interface. You can see this change in the “Library options” field at the bottom of the Library tab of the Project Settings window.

To retrieve information about an object file or object library in an AcuBench project, right-click the object in the File view and select **Properties**. The information shown here includes the file size and compile date, whether or not the object includes debugging symbols, and the names of any resources (like image files) included in the object. You can also view this information using **cblutil**, as described in **Chapter 19** of this manual.

8 Working with Data at the Project Level

Key Topics

Introduction 8-2
Defining Data Files for Use in Projects and Programs 8-2
Creating Data Layout Files 8-4
Working in the File Designer 8-9
Copying DLT Files Between Projects 8-25
Tips for Working in the Data View 8-25

8.1 Introduction

AcuBench includes a set of specialized tools for working with data files and user-defined items. These tools are designed to simplify the creation and maintenance of data-related program elements. Graphical user interfaces, many of which support drag-and-drop editing and syntax checking, make it easy to create, edit, and review data descriptions. At your direction, the information added to these interfaces is used to generate the related code. For an overview of code generation concepts, see **section 3.3**.

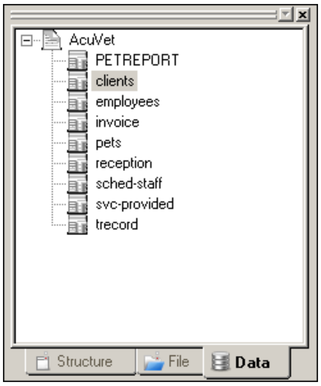
The data-related development tools are collectively referred to as the *data design* tools. The first tool, the File Designer, is accessed through the Data tab of the Workspace window and used at the *project* level to describe the structure of data files used in a project (and to generate FD, SL, SD, and XFD files, as well as file-handling code). The remaining three tools—the Data Set Designer, the Working Storage Editor, and the Linkage Editor—are accessed through the Structure view and used at the program level to define how an individual AcuBench program uses data.

This chapter discusses the process of using the File Designer to define data files for use by any program (or all programs) in an AcuBench project. The program-level data design tools are discussed in **Chapter 10**.

8.2 Defining Data Files for Use in Projects and Programs

As described in **Chapter 3**, programs in AcuBench are organized into *projects*. A project is a group of interrelated programs, some or all of which may make use of a shared set of COPY files, resources, and data files. For this reason, when you set out to define data files for use by AcuBench programs, the first step is to create a definition at the project level.

The project level definition for a data file is created in the File Designer, which is accessed through the Data view of the workspace.



This definition is stored in a data layout file, or DLT. The DLT contains information that AcuBench uses to generate SELECT, File Description, and Sort COPY files, as well as extended file descriptions (XFDs). These files can be used by any program within the project for which the DLT has been defined.

The DLT also contains information about whether and how to generate file handling code in those programs that make use of the data file. You can choose to use no generated file handling code, to use default generated code, or to use code paragraphs that you write and store in the DLT.

If you are working with multiple projects that share one or more data files, you only need to define the data layout for each file one time. Once the DLT has been created in one project, you can easily add it to other projects.

The process of defining data files for use at the project level is as follows:

1. In the Workspace window's Data view, use the File Designer to do one of the following:

    - Create a new data layout file from scratch (**section 8.3.1**)

    - Create a new data layout file from existing File Description (FD) and SELECT COPY files (**section 8.3.2** and **section 8.3.3**)

    - Open an existing data layout file, described in a different project

2. Use the File Control, Definition, and Key tabs of the File Designer to create or update the FD, SELECT, and SORT definitions for the file (**section 8.4.1**, **section 8.4.2**, and **section 8.4.3**).

3. Use the IO Handling tab of the File Designer to specify whether AcuBench should generate file handling code for the file (**section 8.4.4**).

   This is also where you enter any custom file handling code that you would like AcuBench to associate with the data file and make available to any program that uses the file.

4. Use the XFD tab of the File Designer to establish any XFD directives that you would like to include in an extended file descriptor (**section 8.4.5**).

5. Close the File Designer, then generate new FD and SELECT COPY files from the new or revised data layout file. To do this, right-click the icon for your new DLT in the Data view and select **Generate FD/SL** from the pop-up menu.

   The COPY files are assigned a ".fd" and ".sl" extension and, by default, placed in the project's FD folder.

Once you have generated your FD and SELECT COPY files, you are ready to define how individual programs in your project will access your data files. This is accomplished using the Data Set Designer, described in **Chapter 10, section 10.2, "Using the Data Set Designer."**

8.3 Creating Data Layout Files

There are two basic methods for creating a new data layout file:

- You can start from scratch, defining your FD, SELECT, and other file information in a blank data layout.

- You can create COPY files from you existing FD and SELECT statements and use these as the basis for creating a new data layout file. There are two tools that you can use to accomplish this: one for creating a single DLT, one for automating the process of creating multiple DLTs.

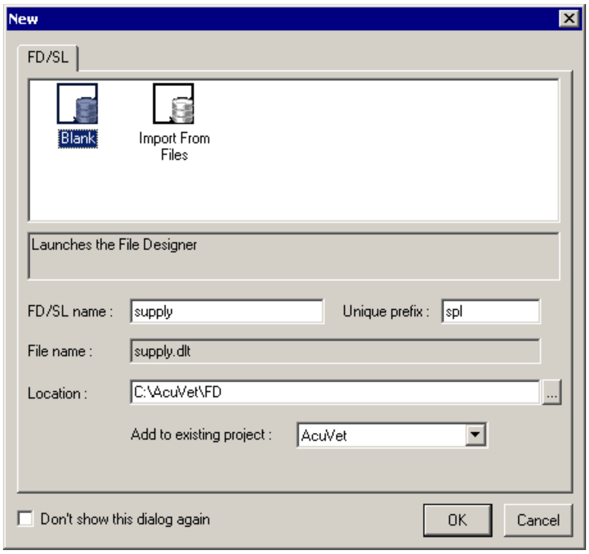
If you are starting from existing code but do not already have ".fd" and ".sl" COPY files, they are easily created by extracting the code from your program and placing the file description and SELECT statement code in separate files. The primary advantages of doing this are:

- You can use the File Designer to maintain the file description and SELECT statement.
- You can designate the ".fd" and ".sl" files as a data set for the program (via the Data Set Designer), which enhances the functionality of the Screen Designer and Report Composer interfaces and causes standard file handling code to be generated (see **section 10.2**)

8.3.1 Creating a DLT from Scratch

To create a brand new data layout file, do the following:

1. Select **New** from the File menu and click on the **FD/SL** tab or right-click the Project node in the Data view and select **New FD/SL** from the pop-up menu.



Launches the File Designer

FD/SL name : supply

Unique prefix : spl

File name : supply.dlt

Location : C:\AcuVet\FD

Add to existing project : AcuVet

Don't show this dialog again

The default FD/SL name, unique prefix, and location settings shown in the New interface are based on settings specified in the Tools/Options interface, under Environment/Prefix.

2. Select the **Blank** icon to create a new data layout that will be used to create a new file description and new SELECT statement.
3. In the “FD/SL name” field, enter a descriptive name for your data layout. This will be used as the base name for the new “.fd” and “.sl” COPY files generated by AcuBench.
4. Specify a location for the new DLT. By default, the data layout and the FD and SL COPY files are all placed in the FD folder for the current project. If you specify a different location for the DLT, the FD and SL COPY files are still placed in the FD folder.
5. If you do not want to add the new data layout to the current project, use the “Add to existing project” drop-down box to select another project.
6. When you are finished, click **OK** to create the data layout and open it in the File Designer.

   You are now ready to define your data layout, as described in **section 8.4, “Working in the File Designer.”**

8.3.2 Creating a DLT from a Single FD/SL Pair

If you have COPY files containing FD and SELECT information, you can use the files to create a data layout file, regardless of whether or not the COPY files were created by AcuBench. The COPY file containing FD information must have a “.fd” extension and the COPY file containing the SELECT statement must have a “.sl” extension.

The “.fd” and “.sl” COPY files that you use can reside on any local or mapped drive. If you use COPY files that do not reside in the current project directory, when you generate your project (or the DLT), the newly regenerated COPY files are placed in their original location and will overwrite any files of the same name located in that directory. The DLT, however, is created in the directory specified in the New FD/SL interface (by default, the project’s FD directory).

Use the following steps to create a data layout file from an existing ".fd" and ".sl" pair:

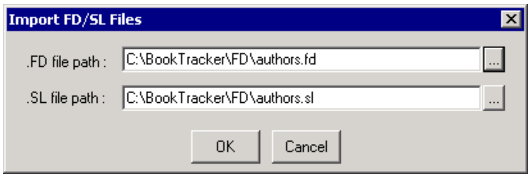
1. Select **New** from the File menu and click on the **FD/SL** tab or right-click the Project node in the Data view and select **New FD/SL** from the pop-up menu.

2. Select the **Import from Files** icon to specify that you have already defined an FD and SELECT statement that will be used to create a new data layout file.

3. Enter a name and location for the data layout file, as described in the previous procedure.

4. Click **OK**.

   The **Import FD/SL Files** dialog box opens, asking you to specify the location of the ".fd" and ".sl" COPY files that you want to use to create the new data layout.

   .FD file path : C:\BookTracker\FD\authors.fd

   .SL file path : C:\BookTracker\FD\authors.sl

5. Enter the path to the COPY files, or use the browse button to the right of the entry fields to navigate to the directory containing each COPY file.

6. Click **OK** to create the data layout and open it in the File Designer.

   You can now make changes to the data layout file, if needed, as described in **section 8.4, "Working in the File Designer."**

Tip: To change the name of an existing DLT, right-click the icon for the DLT in the Data view and select **Properties**. This can be especially useful for the import function, since you enter a name for the DLT before specifying your FD and SL COPY files. Note that changing the name of the DLT does not affect the name of the generated FD and SL files.

8.3.3 Creating DLTs from Multiple FD/SL Pairs

If you have several FD and SELECT COPY files that you would like to bring into AcuBench at the same time, there is an interface to help streamline this process. In order to use this process, your COPY files must have ".fd" and ".sl" extensions, and the COPY files must be added to the project before they are used to create a DLT. The process of adding the files to the project is included in the procedure that follows.

Note that although the COPY files must be added to the project, this can be done without changing their physical location on disk. You have the option to copy your ".fd" and ".sl" files into the AcuBench project directory structure, but this is not required.

It is important to remember that when you generate an AcuBench project or DLT, the new FD and SL COPY files created by the generate process are placed in whatever directory or directories held the original ".fd" and ".sl" files. If you do not want your original COPY files overwritten, it is a good idea to move a master copy to another location, or make a copy of the files within the AcuBench project directory.

The DLT file is placed in the directory specified in the Tools/Options interface, under Environment/Prefix. By default, this is the project's FD directory.

To add multiple ".fd" and ".sl" pairs to an AcuBench project, do the following:

1. Place your ".fd" and ".sl" COPY files in a directory recognized by the Windows operating system. This may be the FD directory of the AcuBench project, another directory on your local hard drive, or a remote directory serving as a Windows mapped drive.

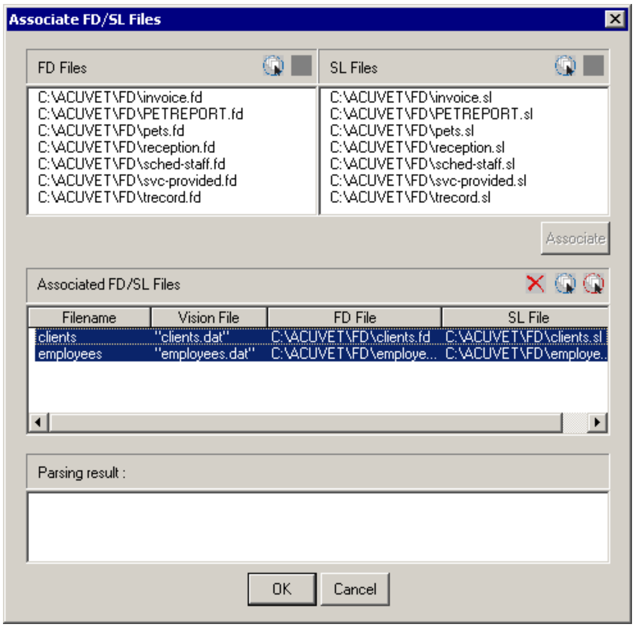
   Remember that when you generate a DLT, newly generated FD and SL COPY files will be placed in the directory or directories that you select in this step. If multiple developers will be using these COPY files, you may want to make the directory or directories read-only, to avoid having one person's changes to a DLT affect everyone's COPY files.

2. In the Workspace window's File view, right-click the FD folder and select **Add/Remove Files**.

3. Navigate to the directory containing your COPY files, select the files you want to use, and click **Add**. Repeat this process for any additional directories containing ".fd" and ".sl" COPY files.

   When you are finished adding files, click **OK**.

4. In the Data view, right-click the project icon and select **Associate FD/SL Files**.



FD Files:

```
C:\ACUVET\FD\invoice.fd
C:\ACUVET\FD\PETREPORT.fd
C:\ACUVET\FD\pets.fd
C:\ACUVET\FD\reception.fd
C:\ACUVET\FD\sched-staff.fd
C:\ACUVET\FD\svc-provided.fd
C:\ACUVET\FD\trecord.fd
```

SL Files:

```
C:\ACUVET\FD\invoice.sl
C:\ACUVET\FD\PETREPORT.sl
C:\ACUVET\FD\pets.sl
C:\ACUVET\FD\reception.sl
C:\ACUVET\FD\sched-staff.sl
C:\ACUVET\FD\svc-provided.sl
C:\ACUVET\FD\trecord.sl
```

Filename	Vision File	FD File	SL File
clients	"clients.dat"	C:\ACUVET\FD\clients.fd	C:\ACUVET\FD\clients.sl
employees	"employees.dat"	C:\ACUVET\FD\employe...	C:\ACUVET\FD\employe...

5. In the Associate FD/SL Files interface, click an FD file and its corresponding SL file, then click **Associate**.

   The pair is added to the "Associated FD/SL Files" list in the middle of the interface.

6. Repeat this process until all of the COPY files have been associated, then click **OK**.

   All of the new DLT files appear listed in the Data View. AcuBench uses the base name of the ".fd" and ".sl" COPY files to create the DLT name. **Section 8.4, "Working in the File Designer,"** describes how to make changes to a DLT, if needed.

8.4 Working in the File Designer

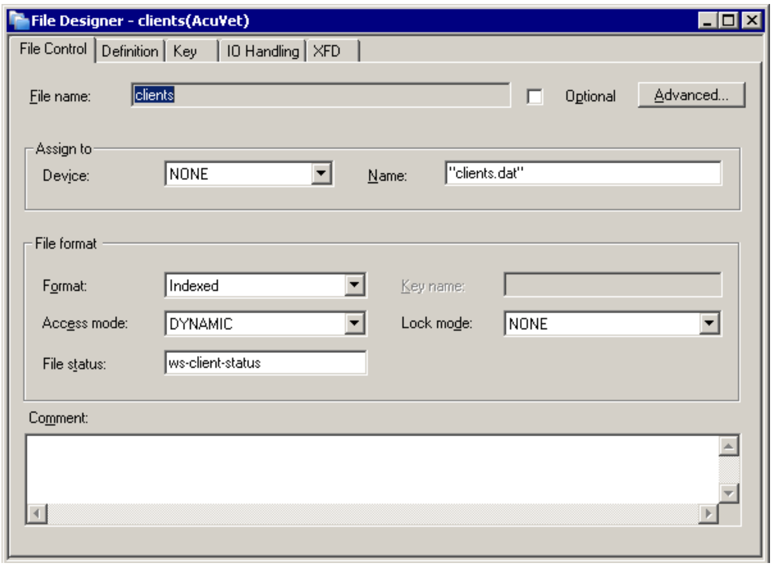
The File Designer, accessed through the workspace's Data view, is used to create, view, and modify data layout files. The designer interface consists of five tabbed screens:

- The File Control tab contains most of the information used to generate a SELECT statement.
- The Definition tab contains the information used to generate a file description (FD).
- The Key tab contains key information for the file (if applicable).
- The IO Handling tab is used to determine which types of file handling code, including declaratives, are associated with a data file at the project level.
- The XFD tab is used to add XFD directives and otherwise customize the XFD generated for a file (when needed).

To open the File Designer, you must either create a new data layout file (as described in **sections 8.3.1** through **8.3.3**) or open an existing data layout file. To open an existing DLT, go to the Data view and double-click the DLT name. You can also right-click on the project node and select **Open *data_layout_name*** from the pop-up menu.

8.4.1 Adding File Control Information

When the File Designer opens, the File Control tab is selected by default. This tab is used to construct most of the SELECT statement that will be generated into the associated ".sl" COPY file. (Key information is entered in a different File Designer tab). Selections made and information entered in this interface must conform to the ACUCOBOL-GT rules for a File-Control paragraph SELECT statement (see Chapter 4 of the *ACUCOBOL-GT Reference Manual*).



File name: clients
Optional: ☐
Device: NONE
Name: "clients.dat"
Format: Indexed
Access mode: DYNAMIC
Lock mode: NONE
File status: ws-client-status
Comment:

Use the File Control tab to view or edit the following information:

1. The File name field displays the name of the current data layout file. The file name is established when you create a new data layout and cannot be changed here.

   To change the name, right-click the DLT in the Data view and select **Properties**.

2. The Optional check box is used to indicate whether the SELECT statement should contain an OPTIONAL phrase. If this box is marked and a program invoking the data file cannot locate the file, a new, empty data file is created with the characteristics defined in the FD and SELECT.

3. The Device and Name fields are used to create the ASSIGN TO phrase of the SELECT statement. First select a device from the drop-down list, then enter a file name or variable. If name is a literal, such as "clientfile.dat", the file name must appear between quotation marks.

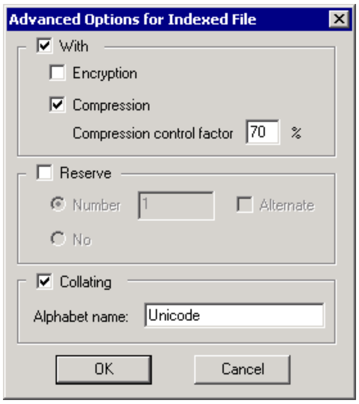
4. Select a file format from the Format drop-down list. Other options in the "File format" area vary depending on the file format that you choose (NONE, Binary Sequential, Line Sequential, Relative, Indexed, or Sort).

   - If you have selected Relative format, the Key name field is enabled. Enter a name for the data item that will hold the value of the relative key.
   - Choose a value from the Access mode drop-down list to determine how the file will be accessed.
   - Select a value from the Lock mode drop-down list to determine what type of file locking is used with this file.
   - Enter a file status variable in the File status field. The value that you enter here is added to Working-Storage automatically.

5. Add a comment in the Comment field, if desired.

6. Click the **Advanced** button to add additional file control information, if needed. The options available in the Advanced dialog depend on the file format that you have specified, and the Advanced button is disabled when you specify a SORT file.

   The Advanced Options for Indexed File screen contains all of the possible options available on the Advanced interface. The interfaces for relative and sequential files contain a subset of these options:


Advanced Options for Indexed File

[✓] With
[ ] Encryption
[✓] Compression
Compression control factor 70 %

[ ] Reserve
● Number 1 [ ] Alternate
○ No

[✓] Collating
Alphabet name: Unicode

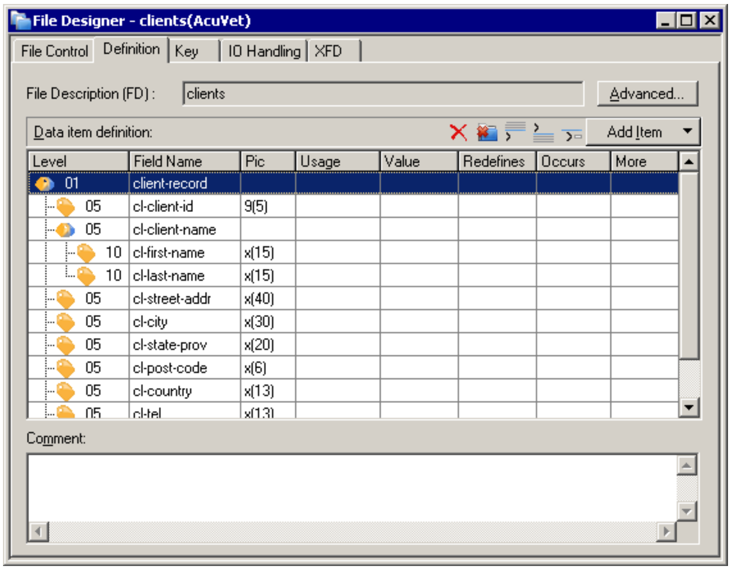
- If you want to enable encryption or compression, mark the **With** check box, then mark the Encryption and/or Compression check boxes. If you enable compression, you must also specify a compression factor.
- To include the RESERVE phrase in your SELECT statement, mark the Reserve check box. You can then either select the No radio button or select the Number radio button and enter a number. Mark the Alternate check box to add an ALTERNATE AREA phrase.
- To include a COLLATING phrase, mark the Collating check box, then enter the name of an alphabet declared in SPECIAL-NAMES in the Alphabet entry field.

To continue creating or editing your DLT, continue with **section 8.4.2, "Adding a File Description."**

8.4.2 Adding a File Description

The Definition screen is used to define the file's record structure. Items entered here are used to create the FD generated into the ".fd" COPY file. Information entered in this interface must conform to the ACUCOBOL-GT rules for a file description entry (see Chapter 5 of the *ACUCOBOL-GT Reference Manual*).

Data items are added, removed, or modified in the “Data item definition” list. Note that you can copy and paste fields from another DLT or from a text file to build a file description in this interface.



File Description (FD) : clients

Data item definition:

Level	Field Name	Pic	Usage	Value	Redefines	Occurs	More
01	client-record						
05	cl-client-id	9(5)					
05	cl-client-name						
10	cl-first-name	x(15)					
10	cl-last-name	x(15)					
05	cl-street-addr	x(40)					
05	cl-city	x(30)					
05	cl-state-prov	x(20)					
05	cl-post-code	x(6)					
05	cl-country	x(13)					
05	cl-tel	x(13)					

To create or modify a file description, select the Definition tab and do the following:

1. To add a field, click either the **Add Item** button or the graphical Add button (the only graphical button enabled when the Data item definition list is empty).

   The graphical Add icon looks like this:

   If you use the graphical Add button, a 01 level item is automatically created. If you use the Add Item button, you can select the level number of your first field, or use items from a COPY file. Linking and importing items from a COPY file is discussed in section 8.4.2.1.

2. Double-click in the Field Name column to edit the default field name, as needed.

3. Click or tab through the remaining columns of the field definition to add PIC, USAGE, VALUE, and other phrases, as needed.

   If you are adding an Occurs phrase, see **section 8.4.2.2** for help navigating the Occurs interface.

   In addition to the basic field information defined in the first seven columns of the "Data item definition" list, you can click in the More column to further refine your field definition. These additional options are discussed in **section 8.4.2.3**.

4. When you are ready to create the next field, you can use the Add Item button, the graphical Add, Add Sub-Item, or Add Item Before buttons.

   In most instances, if you create a field, then want to change its level number, you can double-click in the Level column and type a new value. If the value is not permitted, AcuBench displays an error message. In some instances, the field is read-only, and the level number cannot be changed.

   The default level number assigned to a new item or sub-item can be changed in the Data Designer section of the Tools/Options interface, under Graphical FD.

   Note that you can cut, copy, and paste or drag-and-drop items to make changes to a record.

5. To delete an item, click the graphical **Delete** button. You can also delete all of your field definitions with the **Delete All** button.

6. To specify additional definition information for the file, click the **Advanced** button.

Option	**Description**
Declared	This area lets you select file declaration options.
External	Enable this check box to include the IS EXTERNAL phrase.
Global	Enable this check box to include the IS GLOBAL phrase.
Detailed attribute	This area lets you select file description entry phrases.

Option	Description
Block	Enable this check box to include the BLOCK CONTAINS phrase. Available only for sequential files. Enter the minimum and maximum block size in the entry fields to the right. Enable the appropriate radio button to specify physical record size in terms of either logical records or characters.
Code-Set	Enable this check box to include the CODE-SET phrase. Available only for sequential files. Enter the name of the desired SPECIAL-NAMES character set in the “Name” entry field.
Label	Enable this check box to include the LABEL-RECORDS phrase. This phrase is ignored by the compiler.
Linage	Enable this check box to include the LINAGE phrase. Available only for sequential files. In the entry fields to the right, specify the number of lines on a page, the line number where the footing area begins on the page, and the numbers of lines in the top and bottom margins.
Record	Enable this check box to include the RECORD phrase. Specify the record size in the Record area to the right. Enable the **Contain** radio button to specify the size of a fixed- or variable-length record. Enable **Varying in Size** to specify the size of a variable-length record.
Value of File-ID	Enable this check box to include the VALUE-OF-FILE ID phrase. Enter the file’s external name in the “Name” entry field to the right.
Value of label	Enable this check box to include the VALUE OF LABEL phrase. This phrase is ignored by the compiler.

When you are finished entering File Definition information, you can do one of the following:

- If this DLT describes an indexed file, continue to **section 8.4.3** to define key information.
- To determine what file handling code AcuBench can generate for this data layout, continue to **section 8.4.4**.
- If you want to create a custom XFD for the data file, continue to **section 8.4.5**.
- If you are finished creating your data layout file, close the File Designer, then right-click the icon for your new DLT in the Data view and select **Generate FD/SL** from the pop-up menu. The information in your data layout file is now available to the programs in your project.

8.4.2.1 Linking and importing COPY files

When you use the Add Item button to add fields to your FD, you have the option to link to or import an existing COPY file containing all or some of your field definitions.

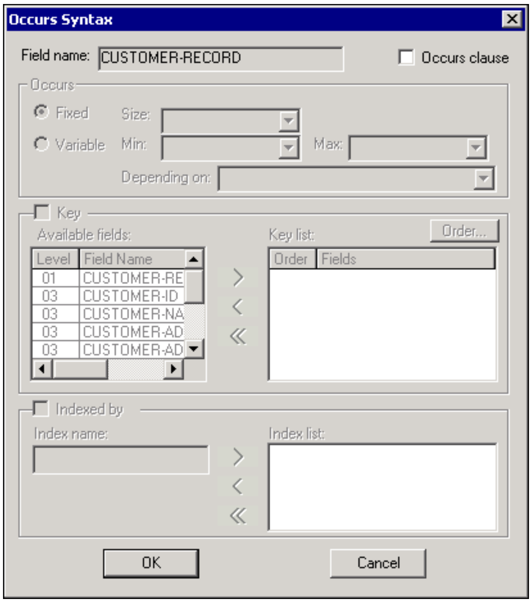
The **Link COPY File** option creates a logical link to an existing COPY file. When you generate your FD and SELECT, a COPY statement is added to the FD to bring in the linked items. Information from the COPY file appears in colored text in the "Data item definition" area. If you link to multiple COPY files, the contents of each file appears in a different color. Linked items cannot be modified in the File Designer.

Alternatively, the **Import COPY File** option makes a copy of the items in the specified ".fd" file. There is no link or dependency on the imported file. Imported items appear in the default text color (usually black) and can be edited in the File Designer.

If part of your record definition includes a link to a COPY file and you want to instead import those fields, you can easily make the switch. To perform the conversion, select a field in the linked portion of the record and double-click in the More field. Click on the button that is displayed. In the Field dialog that appears, enable the **Convert Link to Import** check box and click **OK**.

8.4.2.2 Occurs Syntax dialog

If you want to specify an OCCURS clause for a field, you can either enter a number in the Occurs field or press the browse (...) button to open the Occurs Syntax dialog. This dialog contains graphical interfaces for constructing more complex OCCURS phrases.



The options available in this interface include:

Option	**Description**
Field name	Displays the field name of the selected item.
Occurs clause	Enable this check box to include an OCCURS clause in the data definition.
Occurs	Lets you specify the type of OCCURS phrase.
Fixed	Select this radio button to use a fixed-size OCCURS phrase. Specify a positive integer value in the Size entry field.
Variable	Select this radio button to include a variable-sized OCCURS phrase. Specify a minimum size, maximum size, and DEPENDING ON variable in the adjacent fields.

Option	**Description**
Key	Enable this check box to include the KEY IS phrase.
Available fields	Lists all of the items that can be specified in the Key list. Use the right arrow button to add an item to the Key list.
Key list	Lists all of the selected keys. Remove a selected item with the left arrow. Remove all items with the double-left arrow.
Order	Double-click on an item's Order field to set ASCENDING or DESCENDING.
Order button	Opens the Modify Key Item Order dialog that allows you to change the order of the items in the Key list.
Indexed by	Enable this check box to include the INDEXED BY phrase.
Index name	Enter the name of the index and click the right arrow to place it on the Index list.
Index list	The list of index items. Remove a selected item with the left arrow. Remove all items with the double-left arrow.

8.4.2.3 The Field dialog

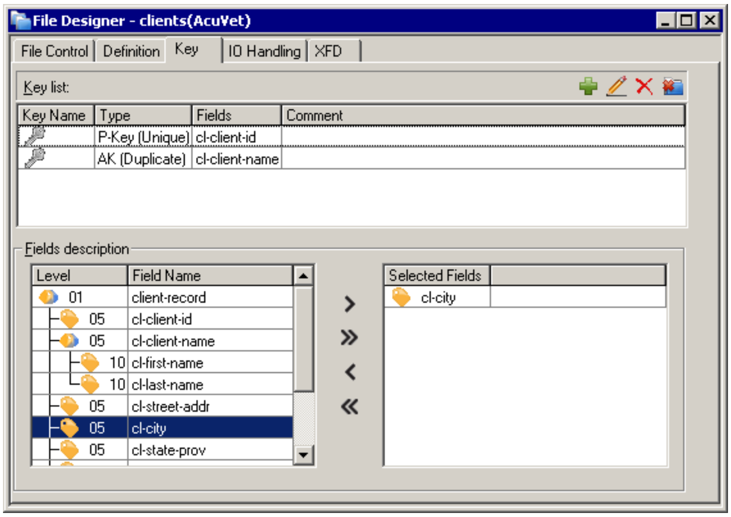
To further refine your field definition, double-click the More column for the field. This opens a Field dialog, which includes the following options:

Option	**Description**
Global	Enable this check box to include the IS GLOBAL phrase.
Special Names	Select a special name type from the drop-down list to include the IS SPECIAL-NAMES phrase.
Sign	Select a sign position from the drop-down list to include the SIGN IS phrase.
Synchronize	Enable this check box and select a position from the adjacent drop-down list to include the SYNCHRONIZED phrase.
Justified right	Enable this check box to include the JUSTIFY RIGHT phrase.

Option	Description
Blank when zero	Enable this check box to include the BLANK WHEN ZERO phrase.
Copy file	This section is enabled only if the selected item is part of a linked COPY file.
Convert Link to Import	Select this check box to convert from a linked COPY file to an import.
Name	Displays the name of the linked COPY file.
Comment	Enter a comment in this field.

8.4.3 Defining Key Information

If you are creating a data layout for an indexed file, you can use the File Designer's Key tab to specify one or more keys. Rules for primary and alternate keys can be found in Chapter 4 of the *ACUCOBOL-GT Reference Manual*.



Key Name	Type	Fields	Comment
	P-Key (Unique)	cl-client-id	
	AK (Duplicate)	cl-client-name	

Level	Field Name
01	client-record
05	cl-client-id
05	cl-client-name
10	cl-first-name
10	cl-last-name
05	cl-street-addr
05	cl-city
05	cl-state-prov

Selected Fields
cl-city

All of the fields that you define in the Description tab appear in the "Fields description" list in the bottom, left portion of the interface. Use this list to build your key(s) as follows:

1. In the “Fields description” list, double-click an item to add it to the “Selected Fields” list. You can also click an item (or Shift-click or Ctrl-click to select multiple items), then click the right arrow (“>”) to add them to the Selected Fields list. To add all fields to the list, click the “>>” button.

   All fields in the “Selected Fields” list when you create a new key are added to the key. If you have mistakenly added a field to the list, use the left arrow (“<”) button to remove it. To clear the “Selected Fields” list, use the “<<” button.

2. To add the fields in the “Selected Fields” list as a key, click the **Add** button (first in the set of four to the right of the “Key list” label). You can also right-click in the Key list area and select **Add**. The selected fields appear in the “Key list.”

   This two-step process for creating keys makes it easier to add split keys, constructed from multiple non-contiguous fields.

3. To add or change a key name, double-click in the Key Name field and enter the name.

4. To change key type (unique primary, primary allowing duplicates, unique alternate key, alternate key allowing duplicates), double-click in the Type field and select an option from the drop-down list.

5. To insert a comment, click in the Comment field and then click on the button that appears.

To change a key for an indexed file:

1. In the “Key list” select the key that you want to change. Note that the key fields of the selected key are listed in the “Selected Fields” list.

2. Add or subtract items from the “Selected Fields” list until you have the fields that you want.

3. Change the key by clicking the **Modify** button (second in the set of four to the right of the “Key list” label).

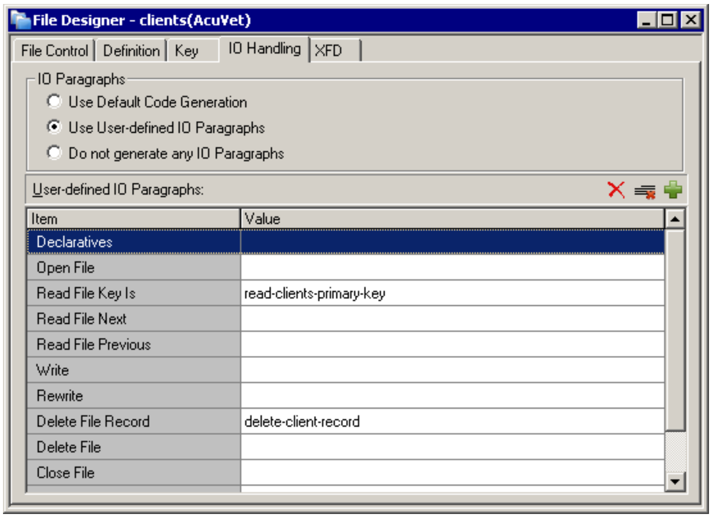
To remove a key, select the key in the “Key list,” then click the **Delete** button (third in the set of four buttons to the right of the “Key List” label). To clear all of the key definitions and start again, use the **Delete All** button (the fourth button in the set).

When you are finished defining and editing key information, you can do any of the following:

- To determine what file handling code AcuBench can generate for this data layout, continue to **section 8.4.4**.
- To create a custom XFD for the data file, continue to **section 8.4.5**.
- If you are finished creating your data layout file, close the File Designer, then right-click the icon for your new DLT in the Data view and select **Generate FD/SL** from the pop-up menu. The information in your data layout file is now available to the programs in your project.

8.4.4 Defining File Handling Behavior

The File Designer IO Handling tab allows you to assign I/O code and declaratives to a data layout file. This interface also lets you determine whether and how I/O paragraphs are generated by AcuBench.


File Designer - clients(AcuVet)

File Control | Definition | Key | IO Handling | XFD

IO Paragraphs
- ( ) Use Default Code Generation
- (●) Use User-defined IO Paragraphs
- ( ) Do not generate any IO Paragraphs

User-defined IO Paragraphs:

Item	Value
Declaratives	
Open File	
Read File Key Is	read-clients-primary-key
Read File Next	
Read File Previous	
Write	
Rewrite	
Delete File Record	delete-client-record
Delete File	
Close File	

To determine what file handling code, if any, AcuBench will generate, do the following:

1. In the IO Paragraphs area of the screen, select one of the three radio buttons.

    - By default, **Use Default Code Generation** is selected. This causes AcuBench to generate default paragraphs for basic file operations, like OPEN, READ, WRITE, and so on.

    - If you want AcuBench to generate your own, custom file handling paragraphs for one or more file operations, select **Use User-defined IO Paragraphs**.

    - If you want to handle all file operations manually within your code, and do not want AcuBench to generate any I/O paragraphs, select **Do Not Generate Any IO Paragraphs**.

2. If you have selected the first or third radio button, skip to the end of this section.

    If you have chosen to use user-defined I/O paragraphs, continue to step 3.

3. In the User-defined IO Paragraphs list on the bottom portion of the screen, the File Designer displays a list of file operations for which AcuBench normally generates code. Double-click in the Value column next to any of these paragraph descriptions to add a new paragraph.

    You can also define your own new paragraph types. To do this, click the **Add** button (the third of the three buttons to the right of the "User-defined IO Paragraphs" label). Double-click in the Item column to enter a description of the paragraph type, then double-click in the Value column and continue with the next step.

4. Enter a paragraph name. If you have already defined custom paragraphs in this DLT, they will appear listed when you expand the Value drop-down list.

5. Click the browse (...) button to enter the Event Editor.

6. Add code to your paragraph.

7. When you are finished, you can either close the Event Editor to return to the IO Handling tab or select another paragraph type from the Message list at the top of the Event Editor window and create additional paragraphs.

To remove a paragraph that you have added for one of the default file operations, click the paragraph name, then click the **Clear Paragraph** button (the second of the three buttons to the right of the "User-defined IO Paragraphs" label).

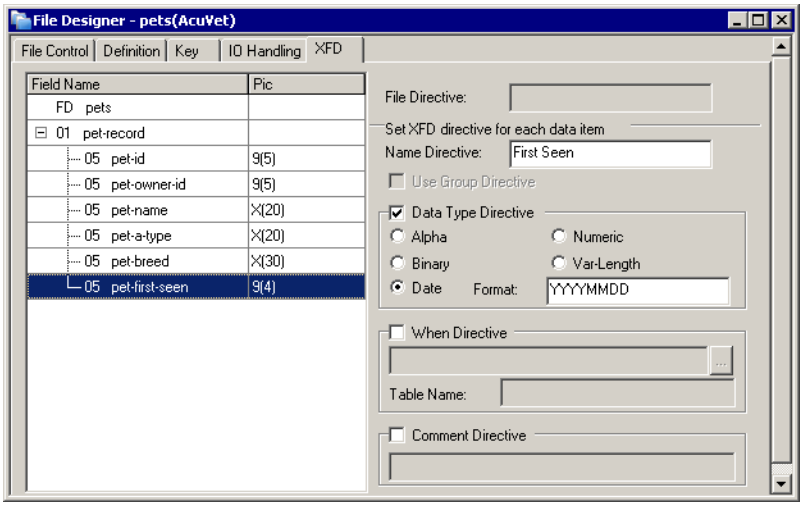
If you have added your own paragraph type and want to remove both the new description and its associated paragraph, select the item and select **Delete** (the first of the three buttons to the right of the "User-defined IO Paragraphs" label).

When you are finished defining file handling behavior in this DLT, you can do either of the following:

- To create a custom XFD for the data file, continue to **section 8.4.5**.
- If you are finished creating your data layout file, close the File Designer, then right-click the icon for your new DLT in the Data view and select **Generate FD/SL** from the pop-up menu. The information in your data layout file is now available to the programs in your project.

8.4.5 Designing a Custom XFD

The File Designer XFD tab lets you add XFD directives to your FD to design an extended file descriptor (XFD) for your data files. XFD files are commonly used to map fields in a record to columns in a table. If you are using AcuXDBC, for example, or reading or writing XML data with AcuXML, XFD files help map the data for translation between file formats. AcuBench can also make use of one XFD directive—the NAME directive—to make some of its Screen Designer and Report Composer interfaces more readable.



Field Name	Pic
FD pets	
01 pet-record	
05 pet-id	9(5)
05 pet-owner-id	9(5)
05 pet-name	X(20)
05 pet-a-type	X(20)
05 pet-breed	X(30)
05 pet-first-seen	9(4)

File Directive:
Set XFD directive for each data item
Name Directive: First Seen
[ ] Use Group Directive
[✓] Data Type Directive
( ) Alpha ( ) Numeric
( ) Binary ( ) Var-Length
(●) Date Format: YYYYMMDD
[ ] When Directive
Table Name:
[ ] Comment Directive

Regardless of whether or not you have made changes to this tab, you can create an XFD file for any data file described in an AcuBench data layout file: right-click the DLT in the Data view and select **Make XFD File**.

General information about XFDs can be found in Chapter 5 of the *ACUCOBOL-GT User's Guide*.

To add XFD directives to your FD, do the following:

1. Select an item in the Field Name list on the left side of the interface.

   The options available on the right side of the interface are enabled or disabled depending on your selection.

2. If you have selected the FD statement, the **File Directive** field is enabled. Enter a base name from which the XFD name will be formed. The default name is based on the ASSIGN TO clause of the SELECT statement.

3. If you have selected a field in your FD, several fields are enabled. Which fields are enabled depends on whether or not the selected field is a group item. Use the information immediately following this procedure to assign directives to data items.

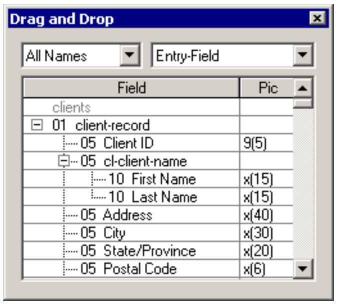
4. When you finish adding XFD directives, close the File Designer, then right-click the icon for your new DLT in the Data view and select **Generate FD/SL** from the pop-up menu. The information in your data layout file is now available to the programs in your project.

   To view the FD (including the XFD directives that you have specified), you can right-click the icon for your DLT in the Data view and select **View *dlt-name*.fd**.

5. To generate an XFD, right-click the data layout icon and select **Make XFD File**. The XFD is generated into the directory containing your FD, SL, and DLT files.

Name Directive

XFD NAME directives can be used to assign a name to a field in a file. This is often used to add a descriptive, formatted, or shortened name to a field, in order to create a more readable table. You can also elect to have alternate names specified with the NAME directive appear in the AcuBench Drag and Drop, Make Radio Button, and Autoload interfaces used to create screen and report controls. These names are then also used as the default title or label associated with the control.



Field	Pic
clients	
01 client-record	
05 Client ID	9(5)
05 cl-client-name	
10 First Name	x(15)
10 Last Name	x(15)
05 Address	x(40)
05 City	x(30)
05 State/Province	x(20)
05 Postal Code	x(6)

To define a new name for a field, enter the name in the **Name Directive** field.

If you want the name specified by the NAME directive to appear in AcuBench interfaces:

1. Open the Tools/Options interface and select the Screen Designer category.

2. Mark the **Apply XFD Names to Drag & Drop** check box.
3. Select the Report Writer category and mark the **Apply XFD Names to Drag & Drop** check box.
4. Click **OK** to save your changes.

By default, data items in the Drag and Drop interface are listed using the field names defined in the file description. For more information about the Drag and Drop interface, see **Chapter 14, section 14.4.2, "Drawing Controls with Drag-and-Drop,"** which pertains to screen design, and **Chapter 17, section 17.4.2, "Using Drag-and-Drop,"** which discusses report design.

Use Group Directive

The USE GROUP directive allows you to enter a group item into the XFD as a single field, instead of using the elements contained in the group. This field is only enabled if you have selected a group item.

You can use the USE GROUP directive in conjunction with the NAME directive to assign a single name for the entire group.

Data Type Directive

When you mark the **Data Type Directive** check box, you can specify a format for the data item other than that established by the PICTURE clause. This means that you can have a database treat a numeric data item as alphanumeric, for example, or have a date item stored as numeric appear in a formatted date format. Select one of the following formats:

- **Alpha** indicates that the selected fields should be treated as alphanumeric.
- **Binary** specifies that the data in the field could be alphanumeric data of any classification. Absolutely any data is allowed.
- **Data** marks the selected data field or group as a date. Enter the data format in the entry field to the right of the Date radio button.
- **Numeric** treats the selected data item as an unsigned integer.
- **Var-Length** designates the data item as having a variable length.

When Directive

The WHEN directive is used when you want to include multiple record definitions or REDEFINES in the XFD file. When you mark the corresponding check box, a field is enabled, allowing you to enter WHEN directive syntax. Click the ellipsis button to display the Expression Builder dialog, where you can also construct WHEN directive syntax.

You can use the **Table Name** field to add an optional tablename clause to the WHEN directive. If you assign a table name, the data that immediately follows the WHEN directive and meets the specified condition will be considered as a separate table.

Comment Directive

You can use COMMENT directives to include comments in your XFD field. Mark the **Comment Directive** check box, then enter a comment in the associated entry field.

8.5 Copying DLT Files Between Projects

Once you have created a data layout file in one project, you can quickly and easily add that DLT to any other AcuBench project, whether or not the other project resides in the same workspace as the original project. It is important, however, to understand the distinction between sharing a DLT between projects and copying a DLT into multiple projects.

When you add a data layout file to a project, you generate new FD and SL files from that DLT. These are generated into the FD folder for the current project, so that if your projects reside in different directories, each has its own set of FD and SL COPY files. The original DLT file, however, is not moved from its original location. This means that if you create a DLT in one project, then add it from another project without making a copy of the original DLT, both projects point to the same file on disk. In this case, if you make a change to the DLT in either project, that change is reflected in both projects.

To open a data layout file defined in a different project:

1. In the Data view, right-click the project node and select **Add FD/SL** from the pop-up menu.
2. In the Add Data Layout to Project window, navigate to the location of the existing DLT. Select the file and click **Open**.
3. Expand the project node (if necessary) and right-click the DLT name to generate new FD and SL COPY files for the project. This makes the information in your data layout available to programs in your project.

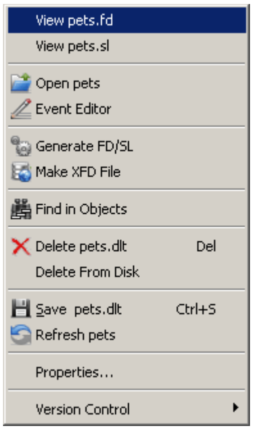
8.6 Tips for Working in the Data View

When you add an existing AcuBench program to a new project, or copy a program from one project to another, it is a good idea to verify that any DLT files referenced by the program exist within the project *before* you attempt to generate the program. Any time a program contains data set definitions, you will need the data layout file and its associated FD and SL COPY files in order to successfully generate and compile that program.

If you generate a program before all of the DLT files that it needs have been added to the project, AcuBench identifies missing, required data fields and adds them to Working-Storage. Later, when you add the missing DLT to the project, you will receive errors caused by the duplicate field names. This can be solved by deleting the duplicate names from Working-Storage, but it's preferable to avoid the problem altogether.

8.6.1 Useful Data View Functions

When you right-click a DLT icon in the Data view, the pop-up menu that appears provides shortcuts to a number of useful functions.



Viewing FD and SL COPY files

You can open and view the FD and SL COPY files generated from the DLT with the **View *base-name*.fd** and **View *base-name*.sl** options, or open the File Designer to view the selected DLT using the **Open *dlt-name*** option.

File handling code and the Event Editor

When you use the IO Handling tab to define file handling code associated with a DLT, that code is edited in the Event Editor. If you are working in the Structure view, however, your file handling code does not appear when you open the Event Paragraph. To edit or add to your file handling code, right-click a DLT in the Data view and select **Event Editor**.

When you launch the interface in this way, the name of the DLT that you have selected appears in the "Control" field at the top of the Event Editor window, and the IO handling paragraphs associated with the DLT appear in the "Message" field. In addition, any file handling paragraphs that you have already created appear in the editor window.

Note that when you use this method to open the Event Editor, code that you have added to the Event Paragraph through the Structure view is hidden.

Creating a new XFD

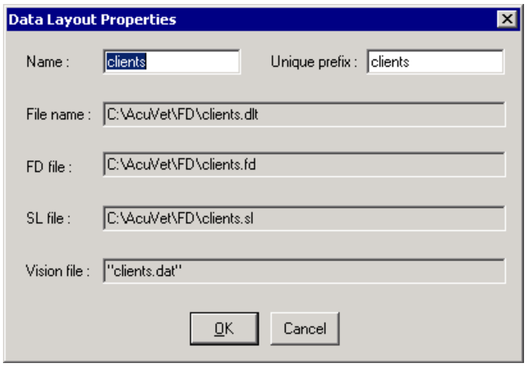
You can create an extended file descriptor (XFD) from a DLT at any time. Just right-click the DLT and select **Make XFD File**. AcuBench invokes the ACUCOBOL-GT compiler in the background to create an XFD for the specified file. The XFD is placed in the project's FD directory.

Refreshing the DLT

If you have made a change to an FD or SL COPY file and want that change to be added to the DLT, select **Refresh *dlt-name***. This option is especially useful if you have created the DLT from your own ".fd" and ".sl" COPY files and are accustomed to maintaining these files outside of AcuBench.

8.6.2 Data Layout Properties

To change the name or unique prefix associated with a DLT, right-click the DLT name and select **Properties**.



Name : clients
Unique prefix : clients
File name : C:\AcuVet\FD\clients.dlt
FD file : C:\AcuVet\FD\clients.fd
SL file : C:\AcuVet\FD\clients.sl
Vision file : "clients.dat"

The Data Layout Properties window also lists the name of the data file associated with the file definition, as well as the path and filename for the DLT file and its FD and SL COPY files. This can be especially useful when you are working with shared DLT or FD/SL COPY files to verify that you are working with the correct version of the correct file. Once the DLT has been created, these paths cannot be changed.

9 Working with Programs

Key Topics

Introduction 9-2
Creating an AcuBench Program 9-2
Adding and Creating Basic Source Files 9-9
Generating a Program 9-16
Compiling a Program 9-16
Executing a Local Program 9-21
Executing a Remote Program 9-21
Debugging a Program 9-22

9.1 Introduction

AcuBench projects can contain two kinds of programs. AcuBench programs have a program structure file and appear in the Structure view of the Workspace window. This type of program contains AcuBench-generated code. Non-AcuBench programs appear as source files, COPY files, and object code only in the File view of the Workspace window. The AcuBench code generator does not generate any code for these programs.

9.2 Creating an AcuBench Program

In AcuBench, creating a program means creating a program structure file (see **section 3.2.3**). After the program structure file is created, you *generate* to create a program source file (".cbl"). The source file can then be modified to include those parts of the program that are not generated by AcuBench.

This section describes the process of creating a new AcuBench program.

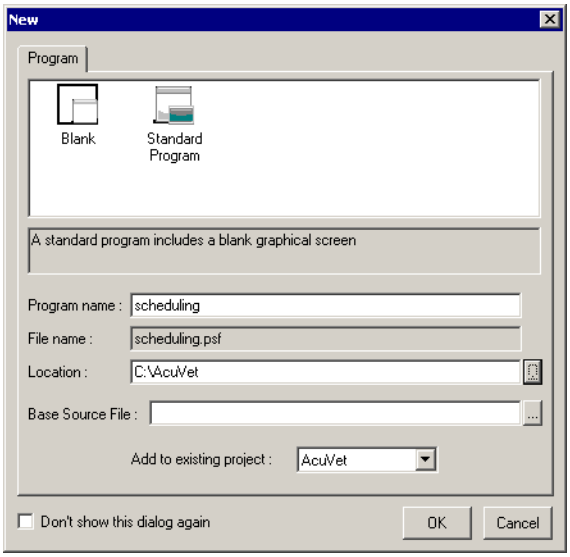
To add an existing program to a project and in the process create a program structure file, see **Appendix 24, section 24.3, "Creating a PSF for an Existing Program."**

To add an existing source file without creating a program structure file, see **section 9.3.2**.

To create a new source file without creating a program structure file, see **section 9.3.1**.

Create a new program as follows:

1. Select **New** from the File menu and click on the Program tab. You can also right-click the project node in the Structure view and select **New Program**.



A standard program includes a blank graphical screen

Program name : scheduling
File name : scheduling.psf
Location : C:\AcuVet
Base Source File :
Add to existing project : AcuVet

2. Select a program template.

   The Standard template includes code for a standard graphical screen.

   The Blank template does not include a screen.

3. Enter the name of the new program in the Program name field.

   The read-only File name field is updated to show the name of the program structure file that will be created for your new program.

4. Enter a location for the program structure file (PSF) in the Location field. To browse for a directory path, click the button to the right of the field.

   By default, the new program structure file is placed in the base directory for the current project.

5. If you want an existing source file to be associated with a new program, enter the file name in the Base Source File entry field or click the browse button to navigate to the file you want to use. The selected source file is renamed to that shown in the Program Name entry field and copied into the source directory of the current project. AcuBench program tags are generated when the program is added to the project. Although program tags are added to the source file, a run-mainscr tag is not generated. You need to create a main screen and the code to call that screen.
6. Select the project to which the new program will be added (this field is set to the current project, by default).
7. Click **OK**.

You should note that when an existing source file is selected in the New/Program dialog, the Code Generation options in its Program/Properties area are selected, and the "Regenerate tagged area only" selection in the Tools/Options/Code Generator/Generate Document dialog should be cleared. Ensure that any automatic code generation capabilities that you don't want are disabled.

If CRT STATUS is declared in your existing source file template, you should ensure that the "Do Not Generate CRT STATUS variable in .wrk" check box in the Program Properties dialog is set.

9.2.1 Adding an AcuBench Program to a Project

When you add an existing AcuBench program to a project, you are actually adding *only* the existing program structure file (PSF) to the project. The COBOL source file (".cbl") and associated COPY files are not added. In addition, adding a PSF to a project creates a pointer, or logical association, between the project and the PSF. It does not move or copy the PSF from its current location into the project directory.

As a result, when you add an existing PSF to a project, consider the following:

- If you have one PSF file in a central location, and that PSF is added to multiple projects, changes made to the program in one project will overwrite changes made to the PSF in another project. In most cases, you should copy a PSF into your project directory before adding it to a project.
- If you have manually added code to the COBOL source file associated with a PSF and want to preserve this code in the new project, you must copy the existing ".cbl" file into the Source folder (or equivalent) for the new project.
- If you have manually added code to the ".cbl" file associated with a PSF and do not want that code to appear in your new project, do not copy the ".cbl" file. In this instance, the PSF acts as a sort of program template from which you can build a new program.

To add a program structure file to a project, use these steps:

1. In the Structure view of the Workspace window, right-click on the project to which you want to add the program and select **Add Program** from the pop-up menu. The Add Program to Project dialog is displayed.
2. In the dialog, locate and specify the name of the program structure file that you want to add and click **OK**.
3. To add the associated program file (".cbl"), use the **Add/Remove** function.

To add a plain source file, use the **Add/Remove Files** function described in **section 9.3.2**.

9.2.2 AcuBench Program Properties

Programs have a variety of properties, including several that specify how code is generated for the program. These properties are set in the Program Properties dialog, which opens when you right-click the program node in the Workspace window Structure view and select **Properties** from the pop-up menu.

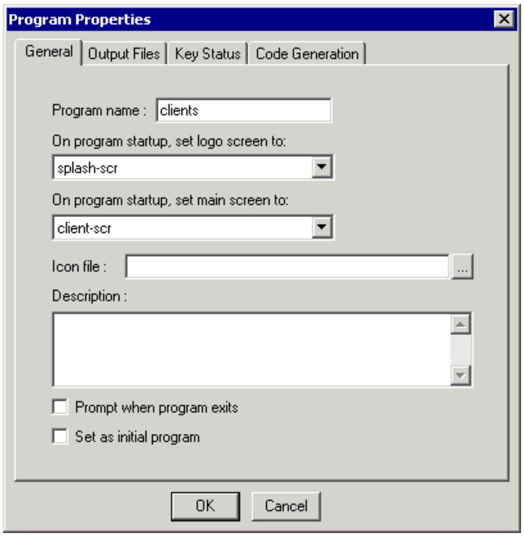
The Program Properties dialog has four tabs: General, Output Files, Key Status, and Code Generation. The General tab includes options for changing the program name, specifying a logo (splash) screen, and assigning an icon to the program, among other properties. The Output Files tab displays the names of the files that are automatically generated for the program and their last modification date (you can control the set of files that are generated for all projects in the workspace in the Tools/Options/Code Generation dialog). The Key Status tab is used to define the program's CRT STATUS variables. The Code Generation tab lets you define the code generation characteristics of a program. These settings override the workspace level settings when that program code is generated, unless the "Follow project default options" check box is set. Detailed descriptions of each tab and their options are given in the next section.

A program options file (".prf") stores default settings for a standard program, shown on the Program Properties Key Status and Code Generation tabs. These settings can be saved as default, saved as another file, or loaded from an existing options file with the "Reference" function.

To set program properties:

1. Right-click the program node in the Structure view of the Workspace window and select **Properties** from the pop-up menu.
2. Review and set values in each tab of the Program Properties dialog.
3. When you are finished, click **OK** to accept the settings and close the dialog.

General Properties



Program name : clients

On program startup, set logo screen to: splash-scr

On program startup, set main screen to: client-scr

Icon file :

Description :

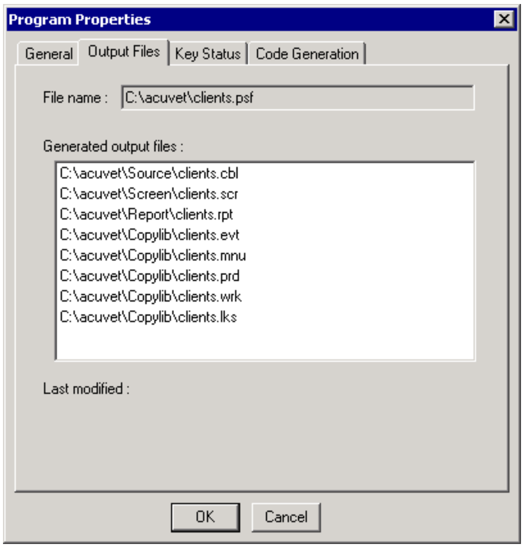
☐ Prompt when program exits

☐ Set as initial program

Option	**Description**
Program name	Specifies the program name. Use this field to change the program name whenever necessary.
On program startup, set logo screen to	Specifies the name of the screen to display as a “splash” screen when the program first starts. If a screen is specified in this field, when you generate the program, AcuBench includes a DISPLAY statement in the ACU-INITIAL-ROUTINE of the “.prd” COPY file.
Display Time	Specifies the amount of time, in seconds, that the logo screen appears on the screen before the main application window is displayed. Takes a value between 1 and 300.
On program startup, set main screen to	Specifies the name of the program’s main screen. All screens that appear in the Structure view for the program are listed; the first screen created in or added to the program is selected by default. AcuBench adds a DISPLAY statement for the selected screen to the ACU-INITIAL-ROUTINE of the “.prd” COPY file.

Option	**Description**
Icon file	Specifies the name of the icon file to be displayed in the upper-left corner of the main window. If a bitmap file is specified in this field, AcuBench includes code to load and display the bitmap in the ACU-INITIAL-ROUTINE of the ".prd" COPY file.
Description	Text entered in this field is placed in the REMARKS section of the program source file (".cbl") between the *{Bench} prgid tags.
Prompt when program exits	Enable this check box to cause the program to prompt for confirmation when the program is closed with the window's close button (in the window's upper right corner). AcuBench generates support code in the program's ".prd" COPY file.
Set as initial program	Enable this check box to designate the program as the main (initial) program.

Output Files Properties



File name : C:\acuvet\clients.psf

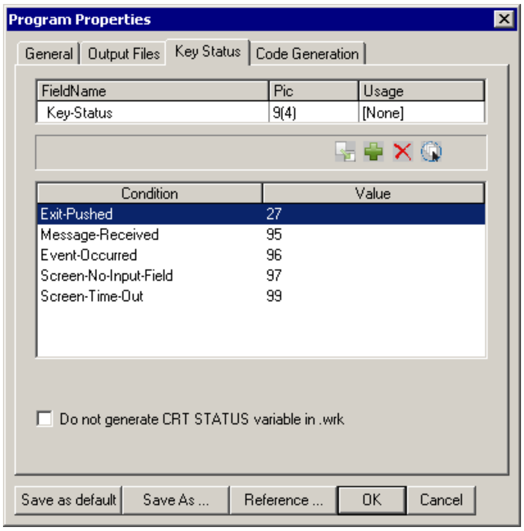
Generated output files :

```
C:\acuvet\Source\clients.cbl
C:\acuvet\Screen\clients.scr
C:\acuvet\Report\clients.rpt
C:\acuvet\Copylib\clients.evt
C:\acuvet\Copylib\clients.mnu
C:\acuvet\Copylib\clients.prd
C:\acuvet\Copylib\clients.wrk
C:\acuvet\Copylib\clients.lks
```

Last modified :

Option	**Description**
File name	Displays the name of the program file.
Generated output files	Lists the names of all files generated by AcuBench for the program. Click on an entry to display the last modification date (the last time the file was generated).

Key Status Properties



FieldName	Pic	Usage
Key-Status	9(4)	[None]

Condition	Value
Exit-Pushed	27
Message-Received	95
Event-Occurred	96
Screen-No-Input-Field	97
Screen-Time-Out	99

☐ Do not generate CRT STATUS variable in .wrk

The Key Status tab is used to define or import CRT STATUS variables. CRT STATUS items are generated into the program's ".wrk" COPY file when the program is generated. Variables defined here are visible to the Screen Designer and other workbench facilities.

Option	**Description**
Field Name (entry field)	Enter the name of the CRT STATUS variable. This name is used in the "*name* IS SPECIAL-NAMES CRT STATUS ..." statement that is generated in the ".wrk" COPY file. It is also used in the EVALUATE statements ("WHEN *name* = *exception_value*") generated in the ".prd" COPY file. This field must contain a value, even if you set the "Do not generate..." check box. The name is set to "Key-Status" by default.
Pic (entry field)	Lets you enter the PICTURE description of the Key-Status variable. Pic9(4) is the default value.

Option	Description
Usage (entry field)	Lets you select a USAGE format from a drop-down list.
Import (button)	Imports level-88 CRT STATUS variables from an existing program. Pressing this button opens a dialog box that lets you locate and select the file containing the variables that you want to import. Note that when you import existing variables from the program's source so that they can be generated into the ".wrk" COPY file, you should later comment out or remove the CRT STATUS variable declarations in the program source so as to eliminate duplicates.
Insert (button)	Displays a dialog for defining a new CRT STATUS item.
Delete (button)	Deletes the selected CRT STATUS item.
Select all (button)	Causes all items on the CRT STATUS list to be selected.
CRT STATUS variable list	Lists all of the currently defined CRT STATUS items. Double-click on any item in the list to edit that item.
Do not generate CRT STATUS variable in .wrk file	Activate the associated check box to disable the generation of CRT STATUS variables in the program's Working Storage (".wrk") COPY file.

Code Generation Properties

The Code Generation tab has the same options as the Tools/Options/Code Generator/Generate Document dialog, and it prefills with the defaults that are set in the Tools/Options interface. Refer to **section 4.6.1, "Generate Document Options,"** for more information about these default settings. If you change any setting in the Code Generation tab after clearing the "Follow project default options" check box, that setting overrides the corresponding Tools/Options setting for that individual program. Setting the "Follow project default options" check box directs AcuBench to use the settings in the Tools/Options/Generate Document dialog. In the latter case, the remaining options on the Code Generation tab are disabled.

Code generation properties also include the following options not included in the Tools/Options interface:

Option	**Description**
Generate source format	This section lets you determine the format for your source file.
ANSI source format	Enable this radio button to have your source file generated in ANSI source format.
Terminal source format	Enable this radio button to have your source file generated in terminal source format.

9.3 Adding and Creating Basic Source Files

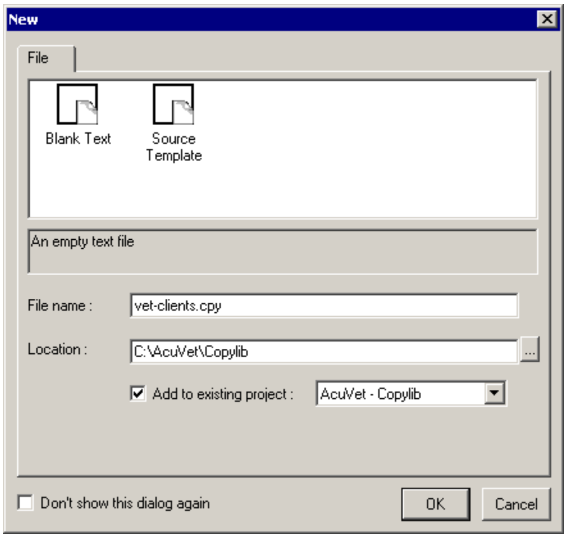
AcuBench provides an interface through which you can add existing source or COPY files to a project or create text-based files of any type. This makes it easy to include new or existing source files that do not include a program structure file in your project. It also provides a way for you to associate notes or code skeletons to a project for reference or documentation purposes.

9.3.1 Creating a File

You can create a source file or plain text file from the File menu as follows:

1. Select **New** from the File menu and click on the File tab, or right-click any folder in the File view of the Workspace window and select the **New File** option. In the latter case, the default file name that appears in the New dialog is automatically given the file extension associated with the selected folder (".scr" for a screen file, for example).

   No matter how you access the New dialog, you are provided with the option of using either of two default templates. If you have added custom templates in the Tools/Options/Environment/Template interface, discussed in **section 4.3.2**, these are also listed here.



An empty text file

File name : vet-clients.cpy

Location : C:\AcuVet\Copylib

[✓] Add to existing project : AcuVet - Copylib

[ ] Don't show this dialog again

2. Select a file template.

   The Source Template includes standard COBOL divisions and sections.

   The Blank Text template is completely empty.

3. Specify a name for the new file. If you used the File/New command to open the New dialog, the default extension is ".cbl".

4. Specify a location for the new file. If you used the File/New command to open the New dialog, the default location is the Source directory for the current project. If you used a File view right-click menu to open the dialog, the default location is the folder on which you right-clicked.

5. Determine whether the file should be added to the current project, another project in the current workspace, or a new project. By default, "Add to existing project" is selected.

6. To add AcuBench tags to the new file and create a program structure file, mark the box next to "Create Program Structure File (.PSF)".

7. When you are finished making changes to the dialog, click **OK** to create the new file.

9.3.2 Adding an Existing Source File

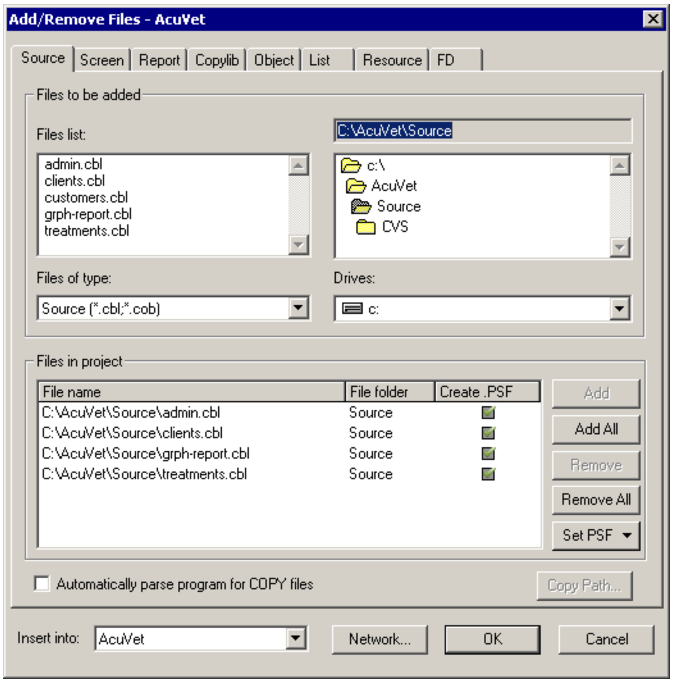
When you add COBOL source files to the project, you have the option of directing AcuBench to automatically search for and add all COPY files named in the source code. To do this, enable the **Automatically parse program for COPY files** check box for each source file being added that you want parsed. After you click **OK**, AcuBench parses the program, identifying COPY files and searching the directories specified in the COPYPATH environment variable to locate them on disk. AcuBench then adds these COPY files to the project, listing the file names in the project's File view Copylib folder. The physical files remain in their original location on disk.

Note: To move a file from one project or folder to another project or folder, simply select it in the File view and drag it to the target folder. A similar cut-copy-and-paste function is not supported.

You can add a source file to a project as follows:

1. Select the desired project and click the **Add/Remove Files** button on the Project toolbar, or select the **Add/Remove Files** option on the Project menu. You can also right-click the project's Source folder in the File view and select **Add/Remove Files**.

This opens the Add/Remove Files dialog with the Source tab selected.



Files list:

admin.cbl
clients.cbl
customers.cbl
grph-report.cbl
treatments.cbl

File name	File folder	Create .PSF
C:\AcuVet\Source\admin.cbl	Source	✓
C:\AcuVet\Source\clients.cbl	Source	✓
C:\AcuVet\Source\grph-report.cbl	Source	✓
C:\AcuVet\Source\treatments.cbl	Source	✓

2. Verify that the “Files of type” field, at the bottom of the uppermost frame in the dialog, shows the correct type of file to be added. Because you are adding a source file, the default extensions are “.cbl” and “.cob” (unless you have established custom settings for the project).

3. From the Drives drop-down list, select the drive on which the file is located.

   You can also use the **Network** button to map a network drive on which your source files are located.

4. In the tree view (above the Drives list, top right), navigate the file system until you open the folder that contains the file to be added.

5. Select (click on) the file to be added in the “Files list”. The file must be highlighted.

To select multiple, non-consecutive files, hold down the **Ctrl** key and click each of the files that you want to select.

6. To add the selected file or files, click **Add**. To add all of the files in the specified directory, click **Add all**. The added files now appear in the "Files in project" list box.

7. To have AcuBench parse the file to automatically add all referenced COPY files to the project, mark the **Automatically parse program for COPY files** check box and modify the COPYPATH environment variable, if necessary (see **section 9.3.3**).

   Parsing is performed after you finish adding files and click **OK** to close the dialog.

8. When you have finished adding files, click **OK** to close the Add/Remove Files dialog.

   The new source file(s) and their associated COPY files (if applicable) are now visible in the File view, and you can use the compiler and runtime settings that you have established for the project to compile and execute the programs. You can also use the Code Editor to view and make changes to your code.

9.3.3 Adding COPY Files

Typically, COPY library files are automatically added to your project when you add source files to the project. To ensure that this happens, make sure that the "Automatically parse program for COPY files" check box is enabled when you add a source file, as described in the previous section.

When AcuBench performs its parsing, it searches for COPY files in the directories defined in the COPYPATH environment variable. Any change to the definition of COPYPATH can affect whether a COPY file is found. A procedure for changing the COPYPATH definition is found in the following subsection.

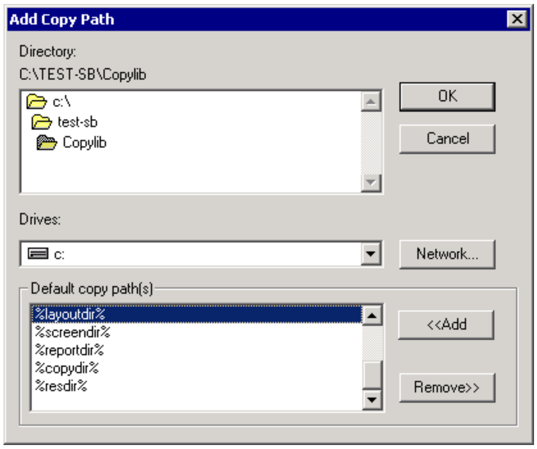
After AcuBench has finished a successful parse, any COPY files that it finds are added to the project's File view Copylib folder. You can reparse a source file at any time by right-clicking the source file icon (in the File view) and selecting Reparse from the pop-up menu.

Adding or removing COPYPATH directories

By default, COPYPATH is set to the relative paths of the project's screen, COPY, resource, and layout (FD) folders, as well as the absolute path to the ACUCOBOL-GT sample directory and its ".def" (COPY files) folder ("C:\Acucorp\Acucbl8xx\AcuGT\sample\def", for example).

To redefine COPYPATH to include other directories, do one of the following:

- If you are working in the Add/Remove Files dialog, click the **Copy Path** button.



  a. Using the mouse, navigate to the desired directory in the Directory list box and then click **Add**. The new path is added to the list of currently defined directories in the "Default copy path(s)" list.

  b. Repeat this procedure for every directory that you want to add to COPYPATH.

  c. When you are done, click **OK**. The Add Copy Path dialog is closed, returning you to the Add/Remove Files dialog.

  d. You can remove a path from COPYPATH by selecting the path from the **Default copy path(s)** list and clicking **Remove**.

- If you are not currently working in the Add/Remove Files dialog, you can change the COPYPATH settings for the project in the Project Settings interface. Select the Environment tab, then follow the steps outlined in **Chapter 7, section 7.5.1, "Working with Environment Variables."**.

Caution: Any change to the COPYPATH definition affects every file in the project. If a directory is removed from the COPYPATH definition and a program in the project has a COPY file located in that directory, that program will fail to compile.

9.3.4 Working with Files in the File View

When you work with files in the File view of the Workspace window, you do not have the option of using AcuBench's code-generating tools, but you can perform typical File menu functions like opening, closing, saving, and deleting files. In addition, you can compile source files (in the project's File view Source folder) and execute object files (in the project's File view Object folder)

Opening a file

To open any file associated with an AcuBench project, go to the File tab of the Workspace window and right-click on the file you want to open, then select **Open**.

Saving a file

To save the contents of the file that has focus, select **Save *filename*** from the File drop-down menu. *Filename* is the name of the file that has current focus. To give another open file focus, either click on the window containing the file or select the file from the Window drop-down menu.

To save the file to another name, select **Save As** from the File drop-down menu and specify a name. If no file extension is specified, the first extension listed in the "Save as type" field is appended automatically.

Closing a file

To close the file that has current focus, select **Close** from the File drop-down menu or click the **Close** icon button located at the far right of the main menu bar.

Deleting a file

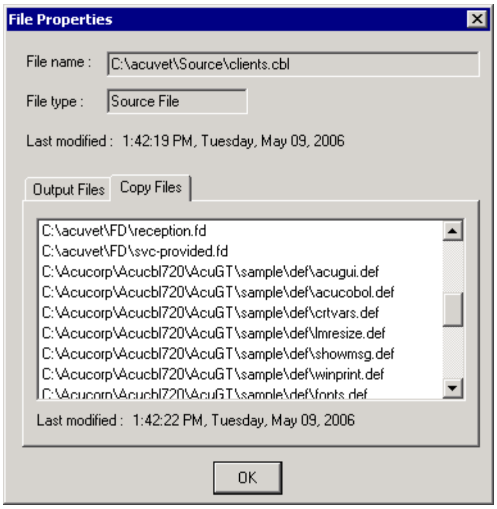
To remove a file from a project, right-click on the file and select **Delete *filename*** from the pop-up menu (*filename* is the name of the selected file). Although the file no longer appears in any Workspace Window view, it is not removed from its location on disk.

To remove a file from the project and delete it from disk, right-click the file and select **Delete From Disk** from the pop-up menu. The file is moved to the Windows Recycle Bin.

9.3.5 File Properties

Files that belong to an AcuBench project have three basic properties: file name, file type, and last modification timestamp. If the file is a program source file, it has a list of output files and a COPY file list. If the file is a remote object file, the dialog contains information about the remote file.

To view file properties, in the Workspace window's File view, right-click on the file of interest and select **Properties** from the pop-up menu.



File name : C:\acuvet\Source\clients.cbl

File type : Source File

Last modified : 1:42:19 PM, Tuesday, May 09, 2006

```
C:\acuvet\FD\reception.fd
C:\acuvet\FD\svc-provided.fd
C:\Acucorp\Acucbl720\AcuGT\sample\def\acugui.def
C:\Acucorp\Acucbl720\AcuGT\sample\def\acucobol.def
C:\Acucorp\Acucbl720\AcuGT\sample\def\crtvars.def
C:\Acucorp\Acucbl720\AcuGT\sample\def\lmresize.def
C:\Acucorp\Acucbl720\AcuGT\sample\def\showmsg.def
C:\Acucorp\Acucbl720\AcuGT\sample\def\winprint.def
C:\Acucorp\Acucbl720\AcuGT\sample\def\fonts.def
```

Last modified : 1:42:22 PM, Tuesday, May 09, 2006

Option	**Description**
File name	Displays the full path and name of the file.
File type	Displays the file type.
Last modified	Displays the file's most recent modification timestamp.

Additional fields for source files (not displayed for remote files):

Option	**Description**
Output Files tab	Lists the name of the compiled object file.
Copy Files tab	Lists all of the COPY files referenced in the file.

9.3.6 Reparsing Source Files

When a source file is added to a project, you can direct AcuBench to parse the source file to identify and add all COPY files referenced by the program to the current project. In the course of development, you may make changes to the source file that introduce new COPY files. When this is true, you should reparse your source file so that the new COPY files are added to the project and found during compilation.

To reparse a file, right-click on the desired source file in the Workspace window's File view, and select **Reparse** from the pop-up menu. When reparsing is complete, a confirmation message is displayed in the Output window.

To reparse all source files in the workspace, select **Reparse All** from the Build drop-down menu. As the reparsing process proceeds, status messages are displayed in the Output window.

Note: AcuBench uses the directories named in the COPYPATH environment variable to look for COPY files. Directories are searched in the order defined (first to last). If the "-Sp" compile option is set on the file, the directories associated with the option are searched to locate COPY files.

The value of COPYPATH is ignored. To check or set the "-Sp" option, right-click the file and select **Program Compile Options** from the pop-up menu. Select **Source Options** from the catalog drop-down list and locate the option labeled "Directories to search for COPY."

9.4 Generating a Program

If a program is displayed in the Structure view of the Workspace window (meaning that it has a program structure file) and you are using workbench tools or options that directly or indirectly specify code (such as the Screen Designer or options in the Program Properties dialog), you need to generate code for the program before you compile the program. Several settings affect what files are created and what code is generated. See **section 3.3, "Automatic Code Generation,"** for a complete discussion of code generation and programming.

To generate code for a program, right-click on the program in the Structure view of the Workspace window and select **Generate Source** from the pop-up menu. Status and error messages are displayed in the Output window.

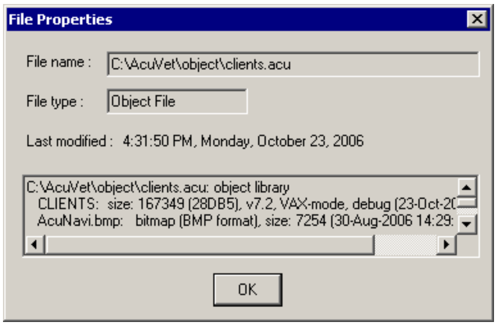
9.5 Compiling a Program

To compile a single, local program, select a program node in the Structure view of the Workspace window, or select a source file from the Source folder in the File view, then right-click and select **Compile**. You can also use the Compile button on the Project toolbar, the Ctrl+F7 keyboard shortcut, or the Build menu's Compile option.

The compile action applies the file-level or project-level settings established in the Project Settings dialog. The compiler command line and any error messages or warnings are displayed in the AcuBench output window.

```
--------- Compiling Program1.cbl ----------
--------- Compile options = -o .\object\@.acu -x -Ga
Program1.acu - Completed: 0 Error(s), 0 Warning(s).
```

After compiling, the compiled object is added to the Object folder in the File view of the Workspace window. You can right-click the File view icon associated with an object file and select **Properties** to retrieve information about the object. This information includes the compiler version, the file size and compile date, whether or not the object includes debugging symbols, and the names of any resources (like image files) included in the object. You can also view this information using **cblutil**, as described in **Chapter 19** of this manual.



File name : C:\AcuVet\object\clients.acu

File type : Object File

Last modified : 4:31:50 PM, Monday, October 23, 2006

```
C:\AcuVet\object\clients.acu: object library
  CLIENTS:  size: 167349 (28DB5), v7.2, VAX-mode, debug (23-Oct-20
  AcuNavi.bmp:   bitmap (BMP format), size: 7254 (30-Aug-2006 14:29:
```

9.5.1 Compiling Programs to a Server

If you are using AcuConnect or AcuServer, you can have AcuBench copy an object file to the server after compiling. There are two ways to do this:

- With the thin client implementation of AcuConnect and the Use Thin Client toggle button.
- With AcuServer or the distributed processing implementation of AcuConnect and the Use Remote Compile toggle button.

Just as there are two methods for placing the object file on the server, there are two ways to specify a remote destination for a compiled object: the Project Properties interface and the Project Settings interface. Settings that you specify in the Project Properties interface are automatically picked up by the Project Settings interface, but the reverse is not true. This means that if you use certain remote settings for testing and a different set of remote settings for production, you can specify the most commonly used set of options in the Project Properties interface, then create a special mode in the Project Settings interface for the less frequently used case.

To use the Project Properties interface to specify remote compiler settings:

1. Right-click the project node in any Workspace window view and select **Properties**.
2. Provide the server name and port on which your AcuConnect or AcuServer listener resides.
3. Specify the directory in which the compiled object(s) should be placed.
4. Mark the appropriate check box next to "Apply to" to indicate whether the settings you have provided should be used at compile time, execution time, or both. Note that execution options only apply if you are using the AcuConnect Thin Client.
5. Click **Test** to verify the connection, then click **OK** to save your changes.

   The next time you compile a program in the project for which you have specified remote compiler settings, the object will be placed on the remote server.

To use the Project Settings interface to specify compiler settings for a specific project mode (or for an individual source file):

1. Expand the Build menu and mark either the **Use Thin Client** or the **Use Remote Compile** toggle button.
2. Right-click the project node in any Workspace window view and select **Settings**.
3. Select the project mode that you want to modify, then select the project or program to which you want to apply the remote settings.
4. In the Standard Options catalog of the Compiler tab in the Project Settings window, enter server, port, and remote directory information.
5. When you are finished, click **OK** to save your settings.

   The next time you compile a program using this mode, the compiled object is placed in the specified server location.

When you use any remote compile option, the object file is listed in the Remote Object folder in the File view. Just as with local object files, you can right-click the object file icon in the File view and select **Properties** to retrieve information about the compiled object.

9.5.2 Compiling Multiple Programs

To compile all of the programs in a workspace, expand the Build menu and select either **Build Workspace** (F7) or **Rebuild Workspace**. The Build Workspace option compiles only those programs that have changed since the last compile operation. Rebuild Workspace compiles every program in the workspace, regardless of whether or not it has changed since the last compile.

If you are using AcuBench code generation, you can use the Tools/Options interface to specify that the Build and Rebuild operations also generate (or regenerate) the programs in the workspace before compiling. To change your Build/Rebuild options, expand the Environment tree and select **Build**.

Note that if you are using remote compile options, you may want to use the Build or Rebuild command twice, once to create local object files and once to create remote object files.

9.5.3 Remote Precompiling with Boomerang

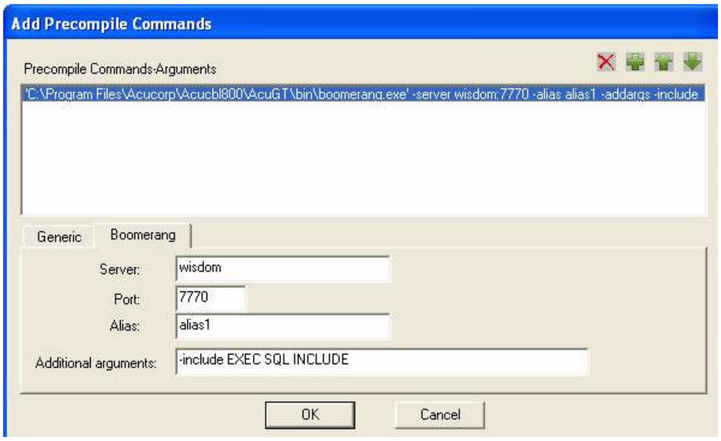
If you have set up and configured Boomerang on a remote server, you can take advantage of an AcuBench interface to the Boomerang client utility. The Boomerang utility sends source files to a remote server for preprocessing, then returns the precompiled source to the local machine for compiling.

For detailed information about the Boomerang utility, including server setup and configuration, see Chapter 3 of the *ACUCOBOL-GT User's Guide.*

To access the Boomerang utility from within AcuBench:

1. Open the Project Settings interface, select the Compiler tab, and open the **Pre-compiler Options** catalog.

2. Mark the check box next to **Generic or Boomerang Precompiler (-Pg)**, then click **Browse** to open the Add Precompile Commands interface.



'C:\Program Files\Acucorp\Acucbl800\AcuGT\bin\boomerang.exe' -server wisdom:7770 -alias alias1 -addargs -include

Server: wisdom
Port: 7770
Alias: alias1
Additional arguments: -include EXEC SQL INCLUDE

3. Select the Boomerang tab, then click the **Add** ("+") button.

4. Use the fields on the tab to enter server, port, and alias information.

   Note that you can specify multiple local and remote precompiler command lines, as needed. The updated precompiler command line appears in the "Precompile Commands-Arguments" field near the top of the Add Precompile Commands window.

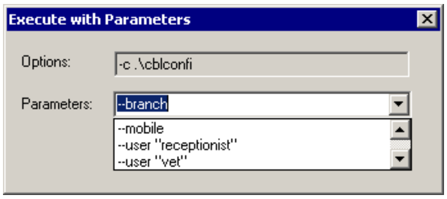
5. If you have added multiple precompiler command lines, you can use the arrow keys to determine the order that the commands will be passed through the compiler.

6. When you are finished, click **OK** to save your changes. The Project Options field at the bottom of the Compiler tab lists the updated compiler command line. Note that you must click **OK** to close the Project Settings window and save your changes. If you click Cancel, any changes to your precompiler settings are lost.

Note that status and any error messages are returned to the output window.

9.6 Executing a Local Program

In the Workspace window, right-click on the program name, program source file, or program object file and select **Execute**. If you have enabled the Allow Parameters option in the Build menu or Project toolbar, this opens the Execute with Parameters dialog box to allow you to specify program parameters.



```
-c .\cblconfi
--branch
--mobile
--user "receptionist"
--user "vet"
```

You can use the Parameters combo box to enter your command line parameters or expand the box to see and select from a list of previously used parameters. The combo box holds up to twenty previous command lines.

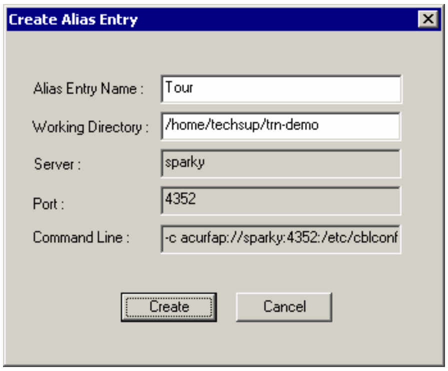
9.7 Executing a Remote Program

If you have established remote settings for your project in order to compile to and run from a remote server using the AcuConnect Thin Client, you can execute your remote programs as follows.

1. Verify that the **Use Thin Client** option is selected. To do this, expand the Build menu or look at the Project toolbar to see that the Use Thin Client button is in “pressed” position.

2. Select the program that you want to execute and select **Execute** from the Build menu. You can also right-click the program and select the Execute command from the pop-up menu, or use the Ctrl + F5 keyboard shortcut.

3. If this is the first time that you have executed the program since your last build, the Create Alias Entry dialog will appear.



Create Alias Entry

Alias Entry Name : Tour

Working Directory : /home/techsup/trn-demo

Server : sparky

Port : 4352

Command Line : -c acurfap://sparky:4352:/etc/cblconf

AcuBench pre-fills the dialog with a default alias name and working directory.

The dialog box also displays server and port information, taken from the Project Properties dialog, and an execution command line, taken from the Thin Client Options set on the Runtime tab of the Project Settings window. This information cannot be changed in this interface.

When you are finished making changes, click **Create** to update the server alias file and execute the program.

9.8 Debugging a Program

To start a program in the workbench integrated debugger, select the program in the Workspace window and select **Go** from the Debug drop-down window. For a complete description of the integrated debugger, see **Chapter 20, "Chapter 20: The AcuBench Integrated Debugger."**

To use the ACUCOBOL-GT runtime debugger, select the **Build/Debug (Runtime)** command or right-click on the program node in the Workspace window File view and select **Debug (Runtime)**. The runtime debugger is described in detail in Chapter 3 of the *ACUCOBOL-GT User's Guide*.

9.8.1 Debugging with the Thin Client

If you are using the AcuConnect Thin Client, you can use the AcuBench integrated debugger or the runtime debugger to debug a program running on the server.

To debug a remote program using the thin client:

1. Verify your thin client settings on the Runtime tab of the Project Settings window (in the **Thin Client Options** catalog), then click **OK** to close the Project Settings interface.
2. Expand the Build menu and select the **Use Thin Client** command.
3. Start the integrated debugger (Debug/Go) or the runtime debugger (Build/Debug Runtime).

   If you have not yet created an alias file entry for the program to be debugged, you are prompted to do so via the Create Alias Entry dialog described in **Chapter 20, "Chapter 20: The AcuBench Integrated Debugger."**
4. Step through the program as usual.

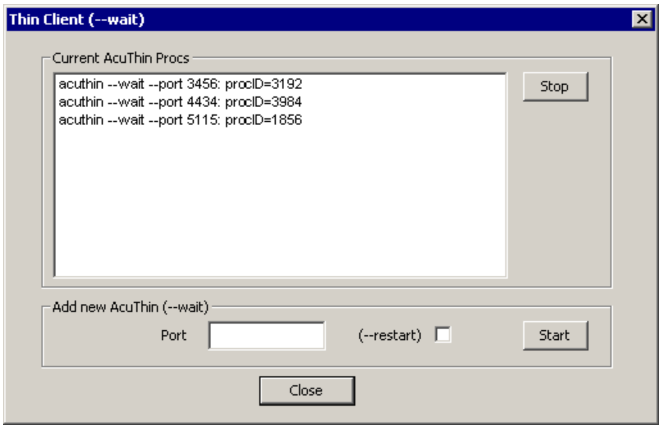
9.8.2 Debugging a Transaction Processing (TP) Application

You can use AcuBench to prepare for debugging in a transaction processing environment, as long as this capability is supported by the vendor of that environment. (Please see your vendor's documentation for details.) You must have Micro Focus's thin client technology installed on the local Windows client and AcuConnect running on the server.

You use a special **acuthin** parameter, "--wait", to start an **acuthin** process on your local machine and have it wait for instructions to open a debugging window. The program executes as initiated through the transaction management environment, not AcuBench. AcuBench provides an interface for enabling this **acuthin** behavior, but does not otherwise interact with the debugging process.

To debug a transaction processing application:

1. Expand the Tools menu and select **Thin Client (--wait)**. This opens the Thin Client (--wait) window.



```
acuthin --wait --port 3456: procID=3192
acuthin --wait --port 4434: procID=3984
acuthin --wait --port 5115: procID=1856
```

2. In the Port field under "Add new AcuThin (--wait)", specify the port where AcuConnect is listening. AcuConnect needs to have been configured to communicate with clients over this port.

3. Select the **(--restart)** checkbox if you want the thin client to restart itself after each debug session. If a transaction requires multiple COBOL programs to run, this allows you to debug them all. You might want to do this if you have pseudo-conversational transactions, for example, where COBOL programs repeatedly complete and then restart. In this case, you would want **acuthin** to continually restart as well.

4. Click **Start** to start an **acuthin** process on the client. The thin client waits for the transaction server to start the runtime.

Note: On UNIX servers, you must set the environment variable A_DEBUG_USING_THIN to a non-zero numeric value before the transaction server can start the runtime with debugging via the thin client. No additional environment settings are required for Windows servers.

5. A runtime debugger screen automatically appears. Debug your application with all the usual runtime debugger options.

6. When you are finished with your debugging session, stop the currently running **acuthin** process (the one in the "wait" state) by selecting the process and clicking **Stop**.

For more information about using the "acuthin --wait" option for debugging, see Chapter 4 of the *AcuConnect User's Guide* and Chapter 9 of *A Guide to Interoperating with ACUCOBOL-GT.*

10 Working with Data at the Program Level

Key Topics

Introduction 10-2
Using the Data Set Designer 10-2
Using the Working-Storage and Linkage Editors 10-10

10.1 Introduction

Chapter 8 introduced the concept of the data layout file, used to describe data files at the project level for the purpose of generating, among other things, file descriptions, SELECT statements, and file handling code. This chapter discusses ways to define data for use at the program level, using the three graphical interfaces: the Data Set Designer, the Working Storage Editor, and the Linkage Editor.

The Data Set Designer is a tool that determines how a data file will be accessed by a given program, and what information from the corresponding data layout file (DLT) will be used by the program. Data sets created with this designer contain a variety of information, including the file open mode, what form of locking to use, and whether AcuBench should generate the code for file operations (such as READ and WRITE).

The Working Storage and Linkage Editors provide graphical tools that you can use to add Working-Storage and Linkage data items to your program. In addition, when you create certain screen and report items in the graphical designers, AcuBench automatically adds needed data items (such as handles) to Working-Storage and displays these in the Working Storage Editor.

10.2 Using the Data Set Designer

The Data Set Designer allows you to identify a set of data files to be used by a program (data sets are associated with programs). Before a data set can be defined, the data files themselves must be defined using the File Designer. When you create a data set, you select from the set of data layout files that are associated with the project.

There are four benefits to creating data sets for your program:

1. The components of each data set are made visible to the Screen and Report Designers. This means that when you want to specify a data item in a control variable, you can select a data file field name from the list of program variables (under the name of the data set).

2. When you generate code, AcuBench generates a set of standard file handling paragraphs for each data set. These paragraphs include OPEN, READ, WRITE, and REWRITE.

3. Via the Event tab of the Data Set Designer's Property window, you can launch the Event Editor to develop BEFORE and AFTER procedures for the data set's associated OPEN, READ, WRITE, and REWRITE routines.

4. You can create associations between fields in data sets such that the field of one data set is marked as *related* to a field in a second data set. When you do this and generate code for the program, a standard *reference* routine is generated that performs a MOVE of the value of the first data item into the referenced (second) data item.

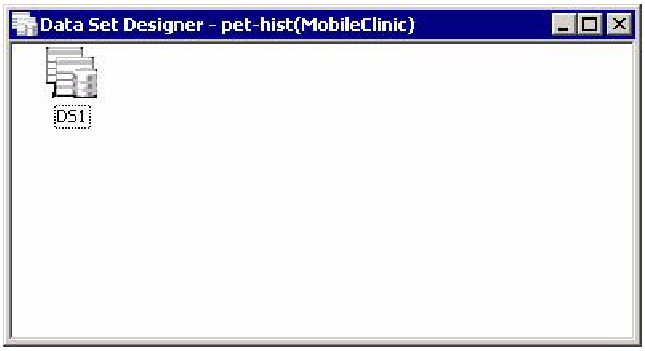
The following sections describe some basic Data Set Designer operations.

10.2.1 Creating a Data Set

Define data for use with an individual program in a project as follows:

1. In the Structure view of the Workspace window, open the desired Program node and right-click on the **Data Set** node (note that data sets are defined for programs, not projects). Select **New Data Set** from the menu.

   A Data Set Designer window is opened with a new data set icon.

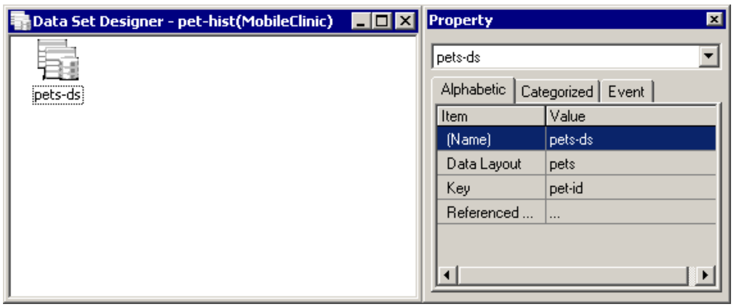


2. Display the Property window by selecting **Property Window** from the View menu.

3. In the Property window, specify a descriptive name for the data set in the Name field, then use the Data Layout drop-down list to select the DLT associated with this data set.

   It is best to assign a data set name that is *different* from your data layout name. If your data layout name is “clients”, for example, you might call your data set “clients-ds”. Both the data layout name and the data set name appear in the Event Editor’s “Message” drop-down list, so assigning different names helps to prevent confusion.

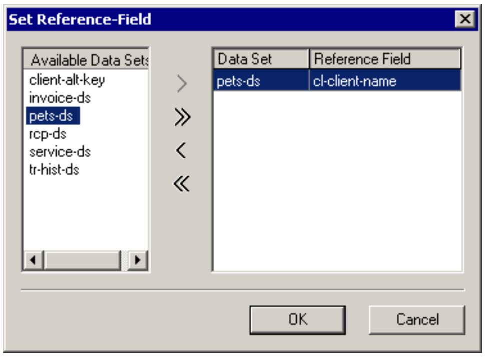
4. If a primary key has been defined in the selected DLT, this value appears in the Key drop-down list. If multiple keys have been defined in the DLT, you can expand the list and select a different key.



Item	Value
(Name)	pets-ds
Data Layout	pets
Key	pet-id
Referenced ...	...

   Because each data set represents a specific manner in which a data file will be used by your program, a DLT with multiple keys may be associated with multiple data sets. In other words, if you have defined more than one key for a file in your DLT, you can define one data set for each key that the program will use to perform file operations.

5. To create an association between a field in the current data set and a field in a different data set, click in the Value column of the Referenced Data Set field and then click the browse (...) button that appears. The Set Reference-Field dialog is displayed.



Select a data set from the list on the left, then click the right-arrow icon to move the data set name to the list on the right. If the default reference field is not correct, click in the Reference Field column and select the correct field from the drop-down list.

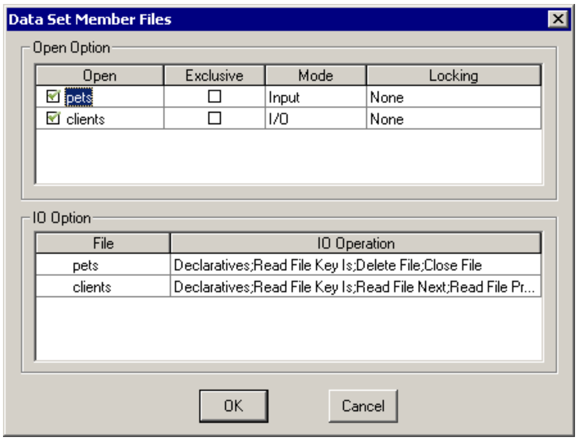
This generates a paragraph similar to the following that you can perform as needed:

```
Acu-cl-ds-Ref-pet-ds.
     INITIALIZE pet-record OF pets
    MOVE cl-client-id OF clients TO pet-owner-id OF pets
     PERFORM Acu-pet-ds-Read.
```

When you are finished, click **OK**.

6. Right-click in the Data Set Designer window and select **Referenced FD/SL Files** to open the Data Set Member Files interface, discussed in **section 10.2.5**

:


Data Set Member Files

Open Option

Open	Exclusive	Mode	Locking
☑ pets	☐	Input	None
☑ clients	☐	I/O	None

IO Option

File	IO Operation
pets	Declaratives;Read File Key Is;Delete File;Close File
clients	Declaratives;Read File Key Is;Read File Next;Read File Pr...

Here, you can determine whether or not AcuBench will generate OPEN code for each data file associated with a data set in your program. You can also determine an open mode (Input, I/O, Extend, or Output), specify a locking mode, and determine what data handling code will be generated within this program.

By default, the type of file handling code that AcuBench generates is based on the open mode that you select. If you select Output mode, for example, AcuBench's default behavior is to generate WRITE code but not READ code. Expand the IO Operation drop-down list for each file to select or de-select code paragraphs to be generated.

When you are finished, click **OK**.

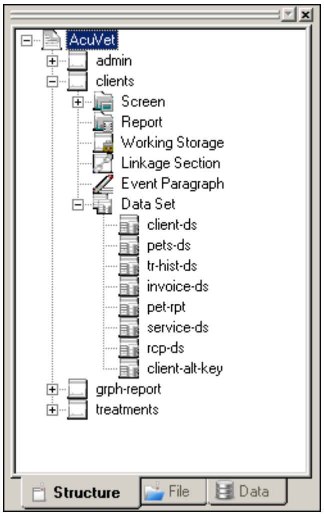
7. Use the Property window's Event tab to specify or create (via the Event Editor) BEFORE and AFTER procedures as needed for each file handling operation (DELETE, READ, REWRITE, or WRITE) associated with the data set.

8. When you are finished, generate the program. The file handling code that you have specified is generated, and the data items described in the data layout file are made available to the other AcuBench design interfaces (for use, for example, with tools like the Screen Designer and Report Composer Drag-and-Drop interface).

10.2.2 Opening an Existing Data Set

You can open an existing data set in the Data Set Designer as follows:

1. In the Structure view of the Workspace window, open the desired Program node, then expand its **Data Set** node to see a list of available data sets.



2. Double-click either the Data Set node or any of the listed data sets. The Data Set Designer opens, showing an icon for each data set.

3. Select the desired data set icon.

4. Display the Property window by selecting **Property Window** from the View menu.

5. View or modify the data set's properties as desired.

10.2.3 Creating a BEFORE or AFTER Procedure

You can use the AcuBench Event Editor to code BEFORE and AFTER procedures for file operations. By default, this code is generated in the program's ".evt" COPY file.

To add code to be executed before or after file operations on a particular file, do the following:

1. In the Structure view of the Workspace window, open the desired Program node and double-click the **Data Set** node. The Data Set Designer window opens, showing an icon for each data set.

2. Select the desired data set, then click the Event tab of the Property window.

   If the Property window is not visible, select **Property Window** from the View menu, or click the Property Window button on the Standard toolbar.

3. In the Event tab, locate the type of procedure that you want to add in the Item list and click in the adjacent Value field. If you have already written the paragraph that you would like invoked before or after the selected file operation, select that paragraph from the Value drop-down list. Otherwise, click the browse (...) button, accept or modify the name in the Add Paragraph dialog, and click **OK**.

   The program's ".evt" file is opened in the Event Editor and a new paragraph is inserted. Code the rest of the procedure.

   Procedures that you enter here are performed immediately before or after the associated AcuBench file handling paragraph is executed. A Before Read paragraph, for example, is performed immediately before the "Acu-*datasetname*-Read" (or "Read-Next" or "Read-Prev") paragraph, each time the latter paragraph is invoked.

4. When you are finished coding a BEFORE or AFTER paragraph, you can close the Event Editor to return to the Data Set Designer. If you are planning to add more BEFORE and AFTER code, you can select another paragraph type from the Message drop-down list at the top of the design window and continue adding code.

10.2.4 The Data Set's Property Window

The Data Set's Property window is used to define the essential values for the data set. This Property window has the following options:

The drop-down list	Select a data set name from the drop-down list at the top of the window to set that data set's values.
Alphabetic tab	This tab lists the data set's properties in alphabetical order.
	(Name) Enter or modify the name for the data set in this field.
	Data Layout Select a data layout from the drop-down list. Only those data layouts that are defined in the associated project appear on the list.
	Key Select a key for the data layout from the drop-down list.
	Referenced Data Set Click in the Value field and then on the ellipsis button to open the Set Reference-Field dialog (described in step 5 of **section 10.2.1**).
Categorized tab	This tab presents the same options as the Alphabetic tab, but they are grouped into categories.
Event tab	This tab is the access path to the Event Editor.
	Item list The list of BEFORE and AFTER procedures that are typical for a data set
	Value The name of the actual BEFORE or AFTER procedure for this item. To launch the Event Editor, select a procedure from the list, click in the Value field, and click on the ellipsis button that appears.

10.2.5 Generating File Handling Code

When you create a data set in the Data Set Designer, you determine what sort of file handling code AcuBench will generate in a given program. As you work with your program, you can change what code will be generated in the

Data Set Member Files dialog. The Data Set Member Files dialog lists all of the data sets defined for a program, as well as information about the file handling code currently specified for each data set.

To open this dialog, go to the workspace Structure view, open the desired Program node, right-click on the **Data Set** node, and select the **Referenced FD/SL Files** command.

To change the OPEN code generated for the file, do any of the following:

- In the "Open Option" portion of the dialog, mark the check box next to the data file name in the Open column to have AcuBench generate OPEN code for the file. If this check box is marked, an "Acu-Open-*file*" paragraph is generated, by default, into the ".prd" COPY file.

  Remove the check mark to stop generating this code paragraph. If the box is not marked, changing other open option details for the data set has no effect.

- In the Exclusive column, mark the check box to indicate that a file can only be opened by one program at a time.
- In the Mode column, select an open mode (INPUT, OUTPUT, EXTEND, I/O). The default mode is I/O.
- In the Locking column, select a lock mode from the drop-down list. The default locking mode is "None".

To change the I/O handling code generated for a file, do the following:

- In the IO Option portion of the dialog, click a file's **I/O Operation** field, then expand the drop-down list.

  The list shows all of the file operations that can be performed on the file, based on the open mode selected in the top portion of the screen.

  Note that this list is populated based on the file handling operations defined in the file's DLT. This means that if you selected the "Do not generate any IO Paragraphs" radio button on the I/O Handling tab of the File Designer, nothing will appear in the I/O Operation field.

- Mark the check box next to an I/O function to have AcuBench generates a paragraph to perform the selected function. Remove the check mark to cause AcuBench to stop generating code for performing the selected operation.

10.3 Using the Working-Storage and Linkage Editors

The Working-Storage Editor and Linkage Editor each provide a specialized interface for defining and maintaining Working-Storage and Linkage section data items. Though these are separate editors, they have identical user interfaces (similar to the File Designer's Definition tab). Therefore, they are documented together here.

10.3.1 The Working-Storage Editor

The Working-Storage Editor is an interactive, graphical editor for defining and maintaining data items that are local to the program and not components of a data file (that is, Working-Storage items). Use the Working-Storage Editor to add, remove, or modify Working-Storage items. Items defined in this editor are generated into the program's ".wrk" COPY file (*program*.wrk), as are Working-Storage items needed by screen controls and other screen elements. Note that the entire contents of the program's Working-Storage COPY file are recreated by AcuBench every time the program is generated. You should never modify the ".wrk" file directly.

When you add screens, reports, screen controls, and report controls in the Screen Designer and Report Composer interfaces, and when you add a data set to a program, AcuBench generates supporting data items and structures, as needed, into the Working-Storage Section. These items may include handles, value variables, file status variables, and so on.

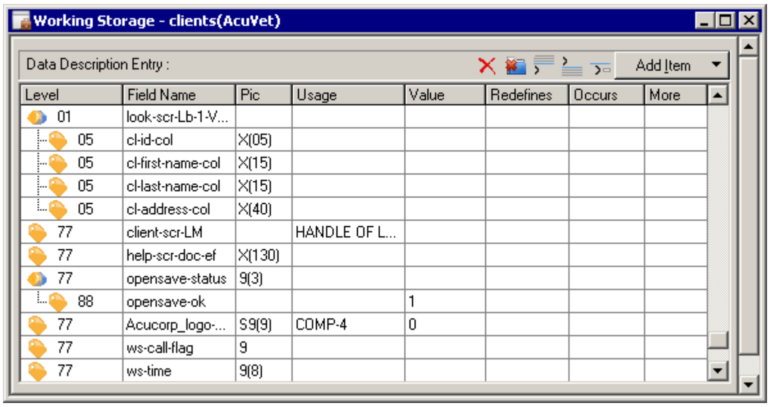
Each Working-Storage item that AcuBench generates is added to the graphical editor. This means that you can modify or delete the generated variables items. Please be aware, however, that changing an AcuBench-generated data item in the Working Storage Editor does not automatically update the corresponding Screen Designer or Report Composer property, so you should exercise caution when making such changes.

10.3.2 The Linkage Editor

The Linkage Editor is an interactive, graphical editor for defining data items that are passed from a calling program (that is, Linkage items). Use the Linkage Editor to add, modify, or remove Linkage section data items. Items defined in this editor are generated into the program's ".lnk" COPY file (*program*.lnk). The entire contents of the ".lnk" file are recreated by AcuBench every time the program is generated. Never directly modify the file.

10.3.3 The Working-Storage and Linkage Editor Interface

To open the Working Storage or Linkage editor, go to the workspace Structure view, open the desired Program node, right-click on the Working-Storage or Linkage node, and select **Open** from the pop-up menu.



Level	Field Name	Pic	Usage	Value	Redefines	Occurs	More
01	look-scr-Lb-1-V...						
05	cl-id-col	X(05)					
05	cl-first-name-col	X(15)					
05	cl-last-name-col	X(15)					
05	cl-address-col	X(40)					
77	client-scr-LM		HANDLE OF L...				
77	help-scr-doc-ef	X(130)					
77	opensave-status	9(3)					
88	opensave-ok			1			
77	Acucorp_logo-...	S9(9)	COMP-4	0			
77	ws-call-flag	9					
77	ws-time	9(8)					

The "Data Description Entry" buttons—from right to left, Delete, Delete All, Add Item Before, Add, Add Sub-item, and Add Item—have the exact same function as the corresponding buttons on the Definition tab of the File Designer. In fact, each of the columns in the Working Storage and Linkage Editor interfaces exactly corresponds to a column in the "Data item definition" list on the Definition tab. Moreover, the Link COPY File and Import COPY File options function the same in both interfaces, and the Occurs Syntax and More dialog boxes are the same. For information, then, about adding data items to the Working Storage and Linkage Editors, please refer to **Chapter 8, section 8.4.2, "Adding a File Description."**

11 Configuring the Code Editor

Key Topics

Introduction 11-2
Establishing Keyboard Shortcuts 11-2
Customizing the Code Editor Interface 11-4
Setting Print Layout Options 11-10

11.1 Introduction

The Code Editor is a highly configurable tool, intended to make working with code as straightforward and comfortable as possible for the COBOL developer. In order to make the tool work for you, you can establish your own keyboard shortcuts for any Code Editor function, change the appearance of the editor, and configure the behavior of various editor functions. For the most part, this configuration is accomplished through the Tools/Options interface, although some settings are configured in other parts of the workbench.

11.2 Establishing Keyboard Shortcuts

In the Tools/Options interface, under Environment/Keyboard, you can set up keyboard shortcuts that affect all aspects of the AcuBench interface. You can also set up specific keyboard shortcuts for the Code Editor interface.

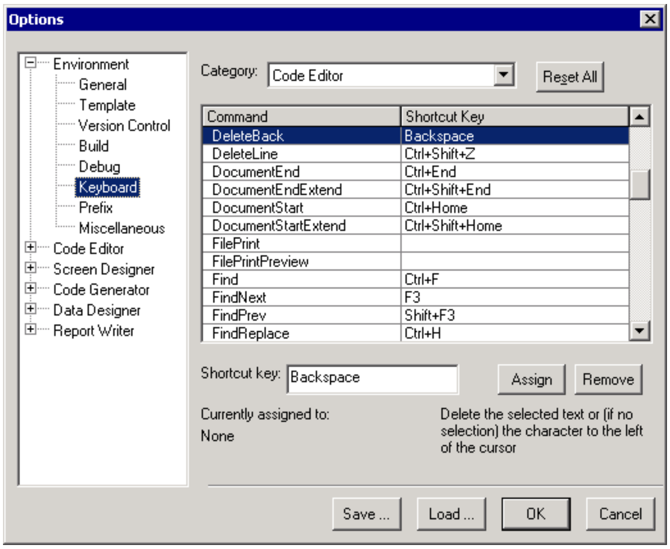
To add, remove, or change a keyboard shortcut associated with a Code Editor function:

1. Expand the Tools menu and select **Options**.
2. Expand the Environment tree and select **Keyboard**.
3. Open the Category drop-down list and select **Code Editor**.

   The list of commands shows only those functions specific to the Code Editor, as well as the keyboard shortcut (if any) currently assigned to each function.

   Note that in some cases, a common function (like Copy or Paste) will appear to have no associated keyboard shortcut. In these cases, the command is associated with a shortcut at the “Main” level. Shortcuts defined at the “Main” level apply across the workbench, unless overridden by a shortcut assigned in a specific context.
4. Select a function in the list, then click in the Shortcut key entry field.

If a keyboard shortcut is currently assigned to the selected function, this shortcut appears in the field. In most cases, a description of the function appears under the "Assign" and "Remove" push buttons.



Category: Code Editor

Command	Shortcut Key
DeleteBack	Backspace
DeleteLine	Ctrl+Shift+Z
DocumentEnd	Ctrl+End
DocumentEndExtend	Ctrl+Shift+End
DocumentStart	Ctrl+Home
DocumentStartExtend	Ctrl+Shift+Home
FilePrint	
FilePrintPreview	
Find	Ctrl+F
FindNext	F3
FindPrev	Shift+F3
FindReplace	Ctrl+H

Shortcut key: Backspace

Currently assigned to:
None

Delete the selected text or (if no selection) the character to the left of the cursor

5. On the keyboard, type the combination of keys that you want to assign to the selected function. The key combination appears in the entry field.

   Note the "Currently assigned to" field under the entry field. If you enter a keystroke that has already been assigned to another function, this field displays the name of that function.

6. To assign the keystroke that you have specified to the function that you have selected, click **Assign**. The new keyboard shortcut appears next to the function name.

7. To remove a keyboard shortcut that has already been assigned to a function, select the function in the Command list, then click **Remove**.

11.3 Customizing the Code Editor Interface

The first four sub-items under Code Editor in the Tools/Options interface provide a way for you to customize the general appearance and behavior of the Code Editor.

11.3.1 Configuring Basic Editor Functions

Configure basic Code Editor functionality in the General interface: the width of the line number pane, the line length associated with text in the Code Editor, the end of line character, and the behavior of the editor's vertical block select function.

You can choose to display a line number pane to the right of the Code Editor window. The number of digits that can be displayed in this pane depend on the pane width. To modify the width of the line number pane from the default (4), enter a new value (from 1 to 6) in the "Line number pane width" text box.

By default, the Code Editor sets the basic line length for editor text at 318 columns. In most cases, if Code Editor text is pasted into another text editor, any trailing spaces are stripped from each line. Some editors, however, preserve the extra spaces, which can lead to line wrapping issues or other undesirable behavior.

To change the default Code Editor line length, enter a value up to 2048 in the "Line Length" entry field. Note that if you open an existing file that contains lines longer than the new limit that you have specified, AcuBench shows a warning message, then wraps any long lines to the new length.

As a Windows editor, the Code Editor's default behavior is to set the end-of-line marker in your files to an ASCII carriage return/linefeed (CR/LF) character. If you plan to save your file in a UNIX environment, you may want to set your end-of-line marker to an ASCII linefeed (LF) character. To toggle between these two end-of-line markers, mark either the **Save Record Delimiter as CR/LF** or the **Save Record Delimiter as LF** option.

The final Code Editor/General option affects two different aspects of the editor. When the "Virtual Space" check box is not marked and you use the vertical block select function (described in **section 12.5.3**), the column is always precisely the width of the last line in the text block. In addition, lines of text created in the editor are *not* padded with spaces to the line length indicated in the “Line length” field.

When “Virtual Space” is selected (the default), you have more flexibility in determining the width of a column selected with the vertical block select function. In addition, all lines in your file are padded with spaces, as needed, to reach the line length indicated in the “Line length” field.

11.3.2 Modifying Editor Appearance

The appearance of the Code Editor window, including the appearance of the text in the window, is configured in the Code Editor/Format section of the Tools/Options interface. This interface allows you to specify two sets of appearance options: one for files opened in terminal format and one for files opened in ANSI format. Note that this interface determines only display characteristics for the different file formats; it doesn’t affect the format of any file.

As you make changes, a small window near the bottom of the Tools/Options interface shows your changes. Use the scroll bar at the bottom of the window to see changes to the Identification Area.

To begin editing Code Editor appearance settings, open the “Source format” drop-down list at the top of the screen and select either **ANSI Format** or **Terminal Format**. Note that the remaining options on the page change depending on which format you have selected.

Modify Code

If you are configuring the ANSI format settings, you can elect to have the Code Editor insert a specified character string in columns 73-80 when a line of code is added or modified. To enable this option, mark the **Modify Code** check box, then enter a string in the adjacent entry field.

Note that inserting the specified string is the *only* task performed by the Modify Code function. Once the string has been placed in columns 73-80, the Code Editor does not maintain the string. This means that if the length of the source line changes, for example, the original modification marker may slip out of the identification area.

Color Options

By default, when an ANSI format file is opened in the Code Editor, the Indicator Area, Area A, and the Identification Area are each assigned a different background color. In addition, comments are shown in green text, reserved words are shown in blue, literal strings are shown in red, and all other text is shown in black.

To disable all color functions, clear the **Enable foreground and background color settings** check box.

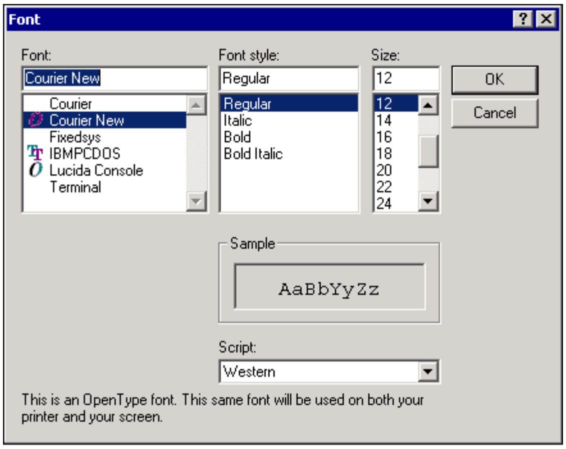
To change the default color settings, do the following:

1. Select an ANSI code area (Sequence Number Area, Indicator Area, Area A, Area B, or Identification Area) or text type (Comment, Keyword, String, Number, or Text) from the list box.
2. The “Background” field displays the current background color associated with the selected item. To change this color, expand the drop-down list and select a color from the list.

   To select a color not on the list, select **User-defined**. This opens a color palette dialog window. You can either select a color in the default palette or click **Define Custom Colors**. When you have made your selection, click **OK**.
3. If it is possible to associate a foreground color with the selected item, the “Foreground” list is enabled. Repeat the process used in the previous step to select a foreground color.
4. If you have selected an ANSI code area, you can change the column width of any area in the “Column” fields. Note that if you change a column setting, you are simply redefining the size of the background color display. You are not redefining any particular area (for example, area A) for the compiler. Columns may not overlap.

Font options

By default, text displayed in the Code Editor appears in the Courier New typeface, with a size of 10 points. To change the display font, click the **Font** button. The Font dialog box appears.



Font: Courier New
Courier, Courier New, Fixedsys, IBMPCDOS, Lucida Console, Terminal

Font style: Regular
Regular, Italic, Bold, Bold Italic

Size: 12
12, 14, 16, 18, 20, 22, 24

Sample: AaBbYyZz

Script: Western

This is an OpenType font. This same font will be used on both your printer and your screen.

Select the font, style, and size that you want, then click **OK**. Once you have clicked **OK** again to close the Options dialog, your text appears in the Code Editor in the new font.

11.3.3 Customizing Tab Stops

By default, when you are working in ANSI format, the Code Editor has tab stops set at columns 7, 8, 12, 73, and 80. When you are working in terminal format, no default tab stops are set. You have the option to add, remove, or change the existing tab stops for each ANSI and terminal format file in the Code Editor/Tabs section of the Tools/Options interface. You can also determine whether pressing the Tab key inserts a tab character or inserts the number of spaces needed to move to the next tab stop.

To customize your tab settings:

1. Open the Tools/Options interface, expand the Code Editor tree, and select the **Tabs** tree item.

2. To determine basic tab behavior, first select one of the two radio buttons near the middle of the screen.

   To preserve the tab character in your source code, select **Keep Tabs**. You can then elect to display a special character to indicate the location of a tab. This can be any non-blank character entered in the "Display tab character with" entry field. The default character is "^".

   To replace tab characters with spaces, select **Insert spaces**.

3. By default, if no tab stops are set, pressing the tab key advances the cursor eight (8) columns. To change this default, enter a number between 1 and 64 in the "Tab size" entry field.

   Note that any tab stops that you have set override the "Tab size" setting. In other words, when you press Tab, the cursor moves to the next tab stop, if any. If there is no tab stop, the cursor moves forward the number of columns indicated in the "Tab size" field. Likewise, when you press Shift+Tab, the cursor moves back to the previous tab stop, or, if there is no tab stop, moves back the number of places indicated in this field.

4. To add, change, or remove tab stops, first select either **ANSI Format** or **Terminal Format** in the "Source format" entry field. You can establish different sets of tab stops for each source format.

5. To add tab stops, enter each desired setting in the "Tabs" entry field and click **Add**. Your new settings appear in the list box. You can set up to 32 tab stops.

   To delete a tab stop, select the setting you want to delete, and click **Delete**. The setting disappears from the list box.

11.3.4 Configuring Keyword Behaviors

By default, the Code Editor displays ACUCOBOL-GT reserved words in blue, while ordinary text is shown in black. In addition to changing the color used to mark reserved words, discussed in **section 11.3.2**, you can add words to a keyword list, or remove words from that list, in order to determine how those words are displayed in the Code Editor. You can also determine how far the next line after the keyword will be indented, if at all.

These keyword lists affect *only* the workbench's colorization and auto-indent functions. Adding or deleting a keyword in a default list does not affect the behavior of the ACUCOBOL-GT compiler. To change the list of reserved words recognized by the compiler, use the "-R" flag, described in Book 1 of the ACUCOBOL-GT documentation set.

You can modify the set of keywords recognized by the AcuBench Code Editor as follows:

- To add an entire keyword set, click **Add**.

  In the Add Keyword Set dialog, enter the name of the new set in the "Keyword set name" text box, then select one of the three radio buttons to indicate the source of the new keyword set. Select **None** to create an original keyword set, **Exist set** to base your new keyword set on an existing set, or **External set** to import a keyword set.

  If you are importing an external keyword set, click the browse button to navigate to the location of the ".ini" file containing the keyword set you want to use.

- To delete a keyword set, select the set that you want to delete in the "Keyword set" drop-down box. Click **Delete**.

- To add or delete a keyword from an existing set, first select a set from the "Keyword set" drop-down box.

  If you are adding a keyword, use the **New** (Ins) push button at the top of the "Keyword" list box to add your new keyword.

  If you are deleting a keyword, select the keyword and click the **Delete** button at the top of the "Keyword" list box.

To determine the automatic indenting behavior associated with Code Editor keywords, select one of the three "Auto indent" radio buttons near the bottom of the dialog.

- If you want the cursor to return to column 1 of the next line after each carriage return, select **None**.

- To have a carriage return send the cursor to the next line at the location of the first non-blank character in the previous line, select **First non-blank token**.

- To use the custom settings in the “Indent value” column of the keyword list to determine cursor positioning after each carriage return, select **Customized**. If the indent value is set to zero, when you press Enter, the cursor indents the next line to the same position as the current line (that is, the behavior described for “First non-blank token” is used). If the indent value is set to a non-zero value, the next line is indented that number of columns beyond the indenting used by the current line.

  To modify keyword indent values, double-click in the “Indent value” column and enter the number of characters to indent the next line.

11.4 Setting Print Layout Options

When you print text files that you have opened in the Code Editor, you can use a standard Windows print dialog to select a printer and manipulate printer settings. In addition to these standard options, AcuBench provides a Page Setup interface that allows you to standardize the layout and formatting used when you print Code Editor documents. You can set a standard header and footer, choose between print styles, determine the default paper and margin sizes, add a frame around the printed text, and specify whether or not to include line numbers in the printout.

To access the Page Setup dialog, expand the File menu and select **Page Setup**.

To see what your document will look like with the settings that you have selected, you can either select **Print Preview** from the File menu, or click the **Print Preview** button on the Project toolbar. The Print Preview button looks like this:

11.4.1 Setting Headers and Footers

By default, when you print a file that is open in the Code Editor, a header and footer are added to each page. The default header prints the file name, centered, and the default footer shows the page number, also centered.

To change the header and footer settings, open the Page Setup dialog and edit the contents of the "Header" and "Footer" entry-fields. You can enter a literal value and/or a combination of special characters, as indicated in the following tables. You can also click the arrow button to the right of the field and select values corresponding to the special characters from a list.

Header and footer values are indicated by the following characters:

Value	**Character**
Full Path Filename	&U
Filename	&F
Page number	&N
Current time	&T
Current date	&D

Alignment positions for header and footer items are represented by the following:

Position	**Character**
Left align	&L
Center	&C
Right align	&R

11.4.2 Setting Page Appearance

In addition to modifying the header and footer information in your printed documents, you can use the Page Setup dialog box to format the appearance of the printed page. You can, for example, choose to print with zebra-style shading, for readability, or to add line numbers.

To make changes to the appearance of the printed page:

1. Open the Page Setup dialog.
2. By default, Code Editor text is printed in simple black and white format. To change this, expand the “Style” list and select another option.

   - If you have a color printer and want your printed output to retain the text color settings used in the Code Editor, select **Color**.
   - To add shading to alternating lines of printed text, select **Zebra**. A light gray shading is used.

3. To add line numbers to the printed text, mark the “Print line number” check box.
4. To add a frame around the printed text, select a style from the “Frame” list box.

   - The “Inner” option, available in all circumstances, places a frame around the text of the file.
   - The “Outer” and “Both” options are available when you have elected to print line numbers. “Outer” places a frame around all body text, including line numbers. “Both” places one frame around the file text and one frame around the line numbers.

5. To change the default document margins, enter the measurements you want in the Left, Right, Top, and Bottom boxes.
6. When you are finished making changes, click **OK**.

   To see what the output file will look like with your changes, select **File/Print Preview**.

12 Working with Source Code

Key Topics

Introduction 12-2
Working with Files 12-2
The Code Editor Window 12-4
Basic Editor Functions 12-5
COBOL-Friendly Editing Functions 12-9

12.1 Introduction

When you double-click a source file or COPY file in the Workspace window's File view, or when you create a new file, the file opens in the AcuBench Code Editor. You can then scroll through the file, type in changes, and use the standard Windows editing functions—search, select all, copy and paste, and so on—to manipulate the text.

In addition to these basic functions, the Code Editor offers a wealth of tools to help streamline code development and maintenance processes. **Color-coded** text and columns, line markers, and custom **tab settings** provide visual cues to reduce typing errors. **Pop-up lists** of program elements, such as variables and paragraph names, help with recall. Editor commands make it possible to select a **vertical column** of text, change the **indentation** of multiple lines of selected code, or instantly **comment** or uncomment an entire block of code. And that's only the beginning.

This chapter begins with a general discussion of working with files in AcuBench, then focuses on the specifics of working with source code in the Code Editor.

12.2 Working with Files

Most of the files associated with your project (listed in the File view of the Workspace window) can be opened in the Code Editor. The obvious exceptions are binary files, like ACUCOBOL-GT object files and bitmaps.

Note that if you are using AcuBench code generation, many of the files in your File view Copylib folder, as well as the files in your Screen and Report folders, are likely to be read-only files. You can open these files in the Code Editor, but are encouraged to make changes in the graphical designers associated with the Structure view.

To open a file listed in the File view, double-click the file name or right-click the file name and select **Open *filename***. The file opens in the Code Editor window, whose title bar shows both the name of the file and the name of the project with which the file is associated.

To save changes to a file, select **Save** from the File menu, click the **Save** icon on the Standard toolbar, or use the default keyboard shortcut: **Ctrl+S**. If you want to save your changes in a new file (or find that you have been working in a read-only file), select the **Save As** option from the File menu, then specify a name and path for the new file.

If you are working with a source file, you can save your changes and compile with the **Ctrl+F7** keyboard shortcut (or right-click the filename in the File view and select **Compile *filename***.

Note that while you can open and save any text file in the Code Editor, you can only compile source files associated with a project in the current workspace. If you try to compile a file not associated with a project, an AcuBench prompt appears asking you whether to add the file to the current project or cancel the compile request.

12.2.1 Creating a New File

When you want to add a new file to a project, the most straightforward method is to right-click a folder in the File view and select **New File**. The file is, by default, given a file extension that corresponds to the file type associated with the folder you select and added to that folder in the project. For example, if you right-click the Source folder and select New File, the file is given a ".cbl" extension and placed in the Source folder. If you select the same command while clicking on the Copylib folder, the default extension is ".cpy" and the file is added to the Copylib folder. You do not have to use the default settings.

At the same time that the new file is added to the project, it is opened in the Code Editor.

By default, two file templates appear in the New File interface: Blank Text and Source Template. The Blank Text template creates an entirely empty file. The Source Template file contains the very basic skeleton of a COBOL program with some error-handling information.

You are not limited to the default templates. In fact, any text file that you create can be used as a template file. Use the Environment/Template section of the Tools/Options interface to add your template file to the New File interface.

12.2.2 File Formats

The Code Editor can work with text in either terminal or ANSI format. By default, the Code Editor makes a clear, visual distinction between files opened in terminal format and files opened in ANSI format. In addition to having code starting in column 1, files opened in terminal format are shown in black and white. In files opened in ANSI format, code starts in column 8, background colors mark the Indicator Area, Area A, and Identification Area, and code text is color coded.

When you have a file open in the Code Editor, you can switch between ANSI and terminal formats. To do this, select **ANSI to Terminal Format** or **Terminal to ANSI Format** from the Format menu. Note that while this changes the appearance of the text in the Code Editor, it does not change the existing spacing or position of text in the file.

12.3 The Code Editor Window

When you open a file in the Code Editor, the text of the file is displayed according to the settings established in the Tools/Options interface under Code Editor/Format. In addition, the editor's title bar lists, by default, the full path and file name of the open file, and a ruler bar indicates each column and tab stop. You also have the option of displaying two additional information areas: one that lists line numbers and one that shows where you've placed bookmarks in your code.

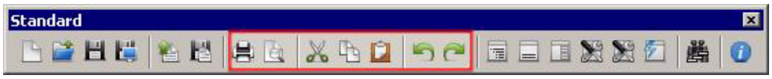
All of these options are configured through the View menu:

- To determine whether the Code Editor title bar displays only the name of the open file or the full path and file name, unmark or mark the **Full Path Filename** menu option.
- To show or hide the ruler bar at the top of the Code Editor window, mark or unmark the **Ruler Bar** option.

- To show or hide a vertical pane listing line numbers for the open file, mark or unmark **Line Number Pane**. The number of digits shown in this pane can be configured through the Tools/Options interface, as described in **Chapter 11, section 11.3.1, "Configuring Basic Editor Functions."**
- To show or hide a vertical pane on which you can place markers to highlight lines of code, mark or unmark **Bookmark Pane**. Use of bookmarks is discussed in **section 12.4.1**.

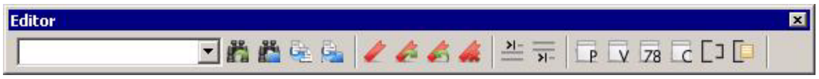
12.4 Basic Editor Functions

When you are working in the Code Editor, you can access all of the standard Windows editing functions through the Edit menu, the Standard toolbar, the Editor toolbar, a right-click pop-up menu, and keyboard shortcuts.



The first group of Standard toolbar buttons let you create a new file, open an existing file, save a single file, or save all files. Other buttons (highlighted in the illustration), allow you to print a file; preview print output; cut, copy, and paste selected text; and undo or redo previous editor actions.

The Editor toolbar provides access to a combination of basic and advanced Code Editor functions.



The first group of Editor toolbar controls lets you enter a search string, find the next instance of a search string, search multiple files for a string, replace a string, or replace a string in multiple files (see **Chapter 21**). Other buttons let you place and navigate between bookmarks (**section 12.4.1**), change the indentation of lines of text (**section 12.4.2**), view lists of paragraphs, variables, and constants declared in the program (**section 12.5.5**), open a COPY file (**section 12.5.6**), and determine the scope of a selected code statement (**section 12.5.4**).

Edit menu commands let you undo or redo previous actions; cut, copy, paste, and delete text; select all text in a file; search for a text string and replace that string, if desired; jump to the a specific point in the file (Edit/Go To); add, remove, and navigate between bookmarks (Edit/Bookmarks); and access a sub-menu of advanced Code Editor functions (Edit/Advanced).

By default, the editor uses the standard Windows keyboard shortcuts for basic functions, such as cut (Ctrl+X), copy (Ctrl+C), paste (Ctrl+V), and select all (Ctrl+A). You can modify these shortcuts, if desired, in the Tools/Options interface, as described in **Chapter 11, section 11.2, "Establishing Keyboard Shortcuts."**

The Code Editor also uses the same navigation keystrokes as many other Windows editors. The arrow keys, for example, can be used to move left or right one character at a time, or up or down one line at a time. The End key moves the cursor to the end of the current line, while the Home key moves you to the beginning of the current line.

Useful navigation keystrokes include:

To move (to):	**Default keyboard command**
Forward one word	Ctrl+Right arrow
Back one word	Ctrl+Left arrow
Top of the file	Ctrl+Home
Bottom of the file	Ctrl+End

Keyboard shortcuts also make it easy to select text without using the mouse. These shortcuts include the following:

Desired Selection	**Keyboard Command**
Next character	Shift+Right arrow
Previous character	Shift+Left arrow
Entire preceding line	Shift+Up arrow
Entire following line	Shift+Down arrow
Preceding multi-line entry	Shift+Up arrow until the desired multi-line entry is selected
Following multi-line entry	Shift+Down arrow until the desired multi-line entry is selected

12.4.1 Using Bookmarks

Bookmarks provide another useful method for navigating through your code. The Code Editor allows you to insert these markers in your source code for reference purposes, then jump forward and backward from maker to marker. These markers do not alter your code. They simply allow you to move around in your file more easily. The Code Editor displays an icon in the Bookmark pane to indicate that a line is bookmarked.

Bookmarks also function as AcuBench debugger breakpoints in your code. More detailed information about breakpoint behavior can be found in **Chapter 20, "Chapter 20: The AcuBench Integrated Debugger."**

Bookmarks are only visible in the Bookmark pane, so that Code Editor window component should be open if you want to see the marks. However, you can still move easily among your bookmarked lines of code, even if you cannot see the marks. Navigation between bookmarks is not affected by icon visibility.

When you work with source files and COPY files in the Code Editor, bookmarks are automatically saved in the workspace file when you close the editor. Note that while you can also use bookmarks in the Event Editor, those bookmarks are ephemeral, lasting only as long as the editor window is open. If your bookmarks vanish unexpectedly, verify which editor you are using. The Event Editor is discussed in **Chapter 14, "Chapter 14: Working with Screens."**

To insert a bookmark in your text, go to the line you wish to mark and click the **Toggle Bookmark** button in the Editor toolbar. You can also expand the Edit menu and select **Bookmark/Toggle Bookmark** or use the **Ctrl+T** keyboard shortcut. Use the same command to remove a bookmark.

When you add a bookmark, if the bookmark pane is visible, a bookmark icon appears next to the selected line. You can show or hide the bookmark pane by marking or unmarking the **Bookmark Pane** check box in the View menu.

To move the cursor to the previous bookmarked line in your code, use the **Previous Bookmark** button on the Editor toolbar. To move the cursor to the next bookmarked line of code, use the **Next Bookmark** button. To clear all the bookmarks from the active document, use the **Clear All Bookmarks** button.

12.4.2 Changing Case and Indenting Lines

The Code Editor allows you to change the case of one or more words or to indent one or more lines without retyping any code.

To capitalize an uncapitalized selection, choose the **Format/Capitalize** command. To change an uppercase selection to lowercase, choose the **Format/Lowercase** command, and to change a lowercase selection to uppercase, choose the **Format/Uppercase** command.

To move the current line to match the indenting of another line of text, do one of the following:

- To change the indent of the selected line to match that of the next line, choose the **Format/Indent to Next** command.
- To change the indent of the selected line to that of the previous line, choose the **Format/Indent to Previous** command.

12.4.3 Viewing Multiple Sections of Your File

A splitter bar makes it possible for you to see (and edit) different parts of your file simultaneously. Each half of the split window has its own scroll bars (horizontal and vertical) for easy movement through your code in either view.

The splitter appears as a small bar above the upper scroll bar arrow on the right side of a Code Editor window. When you drag the bar downward, you open an additional view of the active document, initially at the beginning of the active document. Use the scroll bars or any relocating command in the Edit menu (such as the bookmark commands or the Edit/Go To command) to move to another location in either display.

You can also split your source file window by using the Window/Split command. When you use this command, the cursor changes to a double-arrow that is located at the top of a Code Editor window. Drag the cursor down to open another view of your file.

12.4.4 Merging Data From Another File

While you are working in the Code Editor, you can easily insert the contents of another file into the current document that you are editing. To do this:

1. Position your cursor in the location where you want to insert the file.
2. Choose the **Edit/Advanced/Insert File** command. The Insert File dialog box appears.
3. Locate and select the file you want to insert.
4. Click **Open**. The data appears at the cursor's location.

12.5 COBOL-Friendly Editing Functions

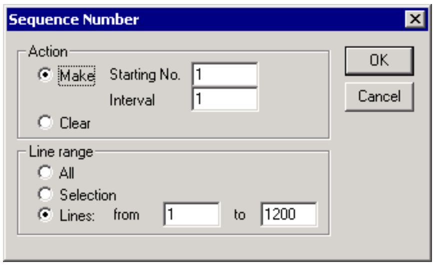
When you work with ANSI format files in the Code Editor, by default, the editing area is divided into five segments: the sequence area, indicator area, Area A, Area B, and identification area. As discussed in **Chapter 11, section 11.3.2, "Modifying Editor Appearance,"** you can determine whether and how these columns are given visual markers in the editor.

12.5.1 Adding and Removing Line Numbers

An additional Code Editor tool can be used to have AcuBench automatically add or modify line numbers in the sequence area. This function is only available when you are working with ANSI format files.

To add line numbers to the sequence area:

1. Open the Format menu and select **Sequence Number**. The Sequence Number dialog opens.


Sequence Number

Action
[✓ Make] Starting No. 1
Interval 1
Clear

Line range
All
Selection
[✓ Lines:] from 1 to 1200

2. To add numbering to your file (or re-number the file), select the **Make** radio button and continue with step 3.

   To clear all existing numbering from your file, select the **Clear** radio button, then click **OK** to return to the editor window.

3. To determine how lines in your file will be numbered, enter a starting number and interval in the appropriate entry fields. By default, the file is numbered from one by one, meaning that the first line in the file is assigned number 00001, the second line is assigned number 00002, and so on.

4. In the "Line range" area, select a radio button to determine whether all lines in the file, a highlighted selection of lines, or a specific line range will be given numbers.

5. When you are finished, click **OK** to number or re-number the file.

12.5.2 Working with Tabs

When you are working in ANSI-format files, the Code Editor, by default, places tab stops at columns 7, 8, 12, 73, and 80. Use the **Tab** key to move the cursor to the next tab stop, or the **Shift+Tab** shortcut to move to the previous tab stop.

As described in **Chapter 11, section 11.3.3, "Customizing Tab Stops,"** you can set up to 32 custom tab stops. You can also set a default tab interval that determines how far the cursor moves when you press the Tab key and no tab stop is set. If you specify a tab size of 3, for example, each press of the Tab key moves the cursor forward 3 spaces. This can be useful for maintaining consistent indenting when working with IF or EVALUATE statements, for example.

Tabs may also be used to move blocks of text in your code. Highlight the lines of code you want to move and press the **Tab** key. The entire block of text shifts to the next tab stop (or to the tab size interval that you have specified).

12.5.3 Working with Blocks of Code

The Code Editor offers several useful tools for manipulating blocks of code. For example, as described in **section 12.5.2**, you can move multiple lines of text with a single press of the Tab key (or the Shift+Tab shortcut). You can also use a single command to add or remove comment markers from an entire block of text, or use a simple key/mouse combination to select, move, copy, and paste a vertical block (or column) of text.

Comment and uncomment block

To comment out a block of code, perform the following steps:

1. Select the text.
2. Expand the Format menu and select **Comment Block**.

   This adds an asterisk character (*) in column 7 and moves each line right one column.

To uncomment a block of code, perform the following steps:

1. Select the text.
2. Expand the Format menu and select **Uncomment Block**.

   This removes an asterisk character and moves each line left one column.

Vertical block select

To select a vertical column of text, position the cursor at the start of the first item in the column, then hold down the **Alt** key while clicking and dragging with the mouse. You can then copy the column of text to the clipboard and paste it to another file, either within AcuBench or in another editor. Note that when you paste a vertical block of text into a third-party editor, AcuBench adds a comment, “**AcuBench Column Block**,” at the head of the column.

When you paste a vertical block into AcuBench, note that the column does not overwrite any existing text. Instead, existing text is moved over the number of columns necessary to make room for the pasted block.

Consider the following example. Suppose you have a list of error codes and the corresponding error messages. You might set up several lines of code like the following:

```
WHEN  ""  MOVE  ""  TO text-message
```

You could then select the error codes in a vertical block, position the cursor between the first set of quotation marks, and paste the block. Each line would be moved to get a result like:

```
WHEN  "01"  MOVE  ""  TO text-message
WHEN  "02"  MOVE  ""  TO text-message
WHEN  "03"  MOVE  ""  TO text-message
```

You could next perform the same operation with the string representing the message to get a result like:

```
WHEN  "01"  MOVE  "File not found      "  TO text-message
WHEN  "02"  MOVE  "Incorrect file type"  TO text-message
WHEN  "03"  MOVE  "File locked         "  TO text-message
```

Note the spaces in the first and third lines. With a vertical block of text, the exact width of the vertical block is preserved, regardless of the length of individual lines in the block.

12.5.4 Using Code Insight Functions

The workbench provides various features to help you keep track of your COBOL code. The first of these, called **code parameters**, provides pop-up tips explaining the syntax and basic usage rules for some COBOL verbs. This feature may be helpful for novice COBOL programmers.

The second code insight function is called **code completion**. When this feature is enabled, you can pause briefly after typing various COBOL verbs to see an alphabetical list of possibilities for completing your statement. For example, if you type:

```
CALL "
```

AcuBench provides a list of programs and library routines that you might want to call. Likewise, if you type “MODIFY” and a space, code completion provides a list of screen controls and handles on which you might want to perform a modify operation. In the latter case, after you choose an item to modify, a second pop-up box provides a list of applicable properties to be modified.

Code and parameter completion are available for the following verbs: CALL, CHAIN, CLOSE, DELETE, DESTROY, GO, INQUIRE, MODIFY, OPEN, PERFORM, READ, REWRITE, START, and WRITE.

By default, code completion is enabled, while code parameters are disabled. You can change these settings in the Tools/Options window. Expand the Code Editor tree and select **Code Insight**. To enable an option, mark the appropriate check box at the top of the interface. To disable and option, remove the check mark. More information about the Tools/Options dialog can be found in **section 4.4.2, "Code Insight Options."**

Two other features can provide additional insight into existing code. If you have code with a good deal of nesting, or code that doesn't use indentation, or if you are having trouble parsing a statement, you can use the **Find Scope** and **Verb Block Match** commands to determine the scope, or range, of a particular statement.

To highlight the block of code corresponding to a specific statement (for example, all lines of code pertaining to a single PERFORM statement):

1. Position the cursor anywhere within the statement or phrase.

2. Select **Find Scope** from the Edit/Advanced menu (or the Editor toolbar).

   AcuBench highlights all lines of code affected by, or included in, that statement.

Similarly, you can use the **Verb Block Match** command to move the cursor from one end of a match verb block to the other. If you position the cursor at the start of a READ statement, for example, and select this command from the Edit/Advanced menu (or Editor toolbar), the cursor jumps to the end of the line containing the corresponding END-READ.

The workbench recognizes the following list of matched verbs as pairs:

Initial	**Final**
Add	End-Add
Call	End-Call
Evaluate	End-Evaluate

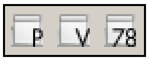
Initial	**Final**
If	End-If
Perform	End-Perform
Read	End-Read
Return	End-Return
Rewrite	End-Rewrite
Search	End-Search
Start	End-Start
String	End-String
Write	End-Write

12.5.5 Using Paragraph, Variable, and Constant Lists

AcuBench maintains lists of the paragraphs, variables, and constants defined in your program. These lists make it easy to jump to the line of code where the paragraph, variable, or constant is defined, or to insert the paragraph, variable, or constant name into your code. When you open a list, it appears in a modeless window that allows you to work in the editor window while the list is displayed.

You can easily access any of these lists while working in the Code Editor or Event Editor using any of the following methods:

- Right-click anywhere in the editor window and select the appropriate command: **List Paragraphs**, **List Variables**, or **List Constants**.
- Open the Edit menu and select **Advanced**, then select the appropriate command: **List Paragraphs**, **List Variables**, or **List Constants**.
- Select the appropriate button on the Editor toolbar:



The lists display the following information:

- The Paragraph List contains the name of each paragraph and the COPY file in which it appears. You can control whether the COPY file column is visible and how the paragraphs are sorted via the Tools/Options/Code Editor/Paragraph List interface.

- The Variable List contains the name, level number, and picture value of each variable, as well as the COPY file in which it appears. You can control which columns are visible, how the variables are sorted, and whether blank FILLER data items are listed in the box via the Tools/Options/Code Editor/Variable List interface.

- The Constant List contains the name and value of each constant, as well as the COPY file in which it appears. You can control which columns are visible and how the constants are sorted via the Tools/Options/Code Editor/Constant List interface.

  Each of the aforementioned Tools/Options interfaces is described in **Chapter 4, section 4.4.3, "Paragraph List, Variable List, Constant List, and COPY File List Options."**

To change the sort order of any list while you are working, click the column heading for the characteristic by which you would like to sort.

You can easily insert into your code any paragraph name, variable, or constant that appears on one of these lists. Position the cursor in your code where you want the item name to appear. In the pop-up list, select a name, right-click, and choose **Paste**. Note that you can change the cursor's position in the code and paste an item multiple times. You can also double-click an item name to insert it at the cursor's location.

Each of the lists offers a variety of additional functions:

- To jump to an item in the list, start typing its name. With each letter you type, the cursor jumps to the first matching item. For example, if you type the letter "a", the cursor jumps to the first item beginning with that letter. If you next type "c" and "u", the cursor jumps to the first (or next) item beginning with "acu". If you mistype, use the backspace key to clear the search buffer.

  When the last item in the list that matches your search string is reached, the search resumes at the top of the list.

- To copy an item name to the Windows clipboard, select the item name in the list, right-click, and select **Copy to Clipboard**.l
- To move the cursor to the line of code where a paragraph, variable, or constant is defined, select the item, right-click, and select **Go to Definition**.
- If you modify your code while the list box is open, you can update the Paragraph List with the **Refresh** command.

12.5.6 Working with COPY Files

AcuBench also maintains a list of the COPY files currently declared in your program. You can view the COPY File List box via the **Edit/Advanced/List COPY Files** command.

The COPY File List functions much like the Paragraph, Variable, and Constants Lists. In addition to the name of each COPY file, the list box contains columns showing each COPY file's directory and the name of the file in which it is declared. You can control which columns are visible and how the COPY files are sorted via the Tools/Options/Code Editor/COPY File List interface, which is described in **section 4.4.3, "Paragraph List, Variable List, Constant List, and COPY File List Options."** You can also determine the sort order by clicking on the column headings in the list box.

To insert a COPY file name into your code, position the cursor in your code where you want the COPY file name to appear. In the COPY File List box, select a name, right-click in the list box, and choose **Paste**. Note that you can change the cursor's position in the code and paste a COPY file name multiple times.

If you are using the COPY File List and want to open a COPY file, just double-click the file name. You can also easily open a COPY file without opening the COPY File List. Just select the COPY statement in your code and do either of the following:

- Select **Open COPY File** from the Edit/Advanced menu.
- Click the **Open COPY File** push button on the Editor toolbar.

The Copy to Clipboard, Go to Definition, and Refresh commands described at the end of the previous section are also available when you are working in the COPY File List.

12.5.7 Using Source Code Templates

You can add an existing source code file to your project as a template through the Tools/Options/Environment/Template interface. The Add function displays the Add New Template File dialog box, with which you locate the source code file you want to use as a template. An icon for your template appears in the “Customize Template for” box in the Tools/Options/Environment/Template interface and in the File/New/File dialog box. After you have added a template to your project, you can modify or delete that file via the Tools/Options/Environment/Template interface. More information about the Tools/Options/Environment/Template dialog can be found in **section 4.3.2, “Template Options.”**

Another source code template function allows you to insert a small section of frequently used code directly into your text. You define these small templates in the Tools/Options/Code Editor/Code Insight interface. Information about defining code templates in the Tools/Options dialog can be found in **section 4.4.2, “Code Insight Options.”**

You can insert a small source code template into your source code as follows:

1. Place the cursor where you want to insert your code template. Choose the **Edit/Advanced/Code Template** command. A list box containing all currently defined code templates appears.
2. Browse the list box for the desired template.
3. Double-click a name; the code template is inserted at the cursor’s location.

12.5.8 Navigating between error lines

When you compile your source files, any compiler errors are listed in the Output window. You can double-click an error in the Output window to jump to the corresponding line in your source file. Once the source file is open in the Code Editor, you can use the Edit/Go To menu to navigate between compilation errors. To go to the previous error line in the Output window error list, expand the Go To submenu and select **Previous Error**. To move the cursor to the next error line, select **Next Error**.

13 Configuring the Screen Designer

Key Topics

Introduction 13-2
Customizing the Screen Designer Interface 13-2
Establishing Screen and Control Defaults 13-4
Adding Screen Templates 13-6
Configuring Keyboard Shortcuts 13-7

13.1 Introduction

As you prepare to design a user interface for your AcuBench program, it is important to ensure that all of the required foundation elements have been put in place. When you use the Standard Project template (which opens with a blank screen in the Screen Designer), it is tempting to jump right in to designing screens. But keep in mind that the process of creating a user interface is significantly easier if you have defined data layout files for your project and created data sets within your program.

With this in mind, before you begin designing and working with your screens, you should:

1. Create an AcuBench project (discussed in **Chapter 6**).
2. Define one or more data layout files in the Data view (discussed in **Chapter 8**).
3. Create a new AcuBench program (discussed in **Chapter 9**).
4. In the Structure view, define one or more data sets for use by the program (discussed in **Chapter 10**).

Once you have completed those four steps, you are ready to start working in the Screen Designer. This chapter describes how to configure the Screen Designer environment, set screen and control defaults, and prepare to start working with screens. **Chapter 14** describes how to use the Screen Designer to create user interfaces for your AcuBench programs.

13.2 Customizing the Screen Designer Interface

The Screen Designer interface, accessed through the workspace Structure view, is made up of a design window and two companion tools. The central design window displays the screen form on which you place controls to create a character-based or graphical user interface for your program. The tools are the Screen Component Toolbox, which displays all of the standard ACUCOBOL-GT screen controls and available ActiveX controls that can be added to your screen form. The Property window displays a configurable list of properties associated with the selected screen or control.

The appearance of the design window and the contents of the Property window are configured through the Screen Designer section of the Tools/Options interface. This section addresses the options for customizing the appearance of the design window. Property window configuration is discussed in **section 13.3**.

By default, when you create a new screen or open an existing screen in the Screen Designer, the screen form appears covered with a grid of black dots. This grid does not directly correspond to any generated COBOL code, but instead acts as a visual aid to help you position and align controls on the screen.

To configure the appearance and behavior of this grid:

1. Open the Tools/Options interface, expand the Screen Designer tree, and select **General**.

2. To change the size of the cells making up the grid, enter a number between 3 and 20 in the "Grid width" and "Grid height" fields. (You can either select the value from the drop-down list or type the entry in the field.)

   By default, the size of each grid cell is set to a width and height of 10 pixels. This corresponds to the default value for the screen CELL SIZE property.

3. If you would like each control that you place on the screen form to automatically align to the grid boundaries, mark the **Snap to grid** check box.

   Note that even when this option is selected, you can still select a control and move it in any direction one pixel at a time by using the keyboard's arrow keys.

You can also configure one additional Screen Designer behavior through this interface. If, when you draw a new control under an existing control in the Screen Designer, you would like the new control's left boundary to align to the left boundary of the existing control, mark the **Auto alignment** check box. This automatic alignment occurs when the boundaries of the two controls are within 5 pixels of each other horizontally.

13.3 Establishing Screen and Control Defaults

In the Tools/Options/Screen Designer interface, the **Default** and **Visibility** screens let you determine not only the default properties given to any new screen or control created in the Screen Designer window, but also which properties appear in the Property window for each screen item. This means that if you want all entry fields to show only uppercase text, and you don't want anyone to change that setting, you can set the default case in the property sheet to "Upper", then remove the "Case" property from the property window entirely.

Certain properties that appear in the property sheet cannot have default values assigned. Location properties, for example, cannot have a default value, because they are dependent on where you position a control on the screen form.

Some properties do not appear on the property sheet at all. For the most part, these properties are not part of the initial state of the control, and therefore cannot be described in the Screen Section. Such properties include the SELECTION-TEXT property, used to determine what text the user has selected within an entry field, and the RESET-LIST property, used to delete the items that currently exist within a list box. Because a user selection occurs after the creation of the control, and a list is only cleared after it has both been created and loaded with data, it wouldn't make sense to set either of these properties as part of the control's Screen Section definition.

For each control property that does appear in the Screen Designer Property window, you can determine the default value as follows:

1. Open the Tools/Options interface, expand the Screen Designer tree, and select the **Default** category.
2. Select an item from the Controls list. This list contains all of the screen items to which properties can be assigned. This includes the screen itself, menus, and toolbars, as well as each of the standard ACUCOBOL-GT controls. In some instances, control sections (such as menu items and tree view items) also have their own configurable property defaults.

   The Properties list (to the right of the Controls list) is updated to show the properties associated with your selection.

3. To change the default value assigned to a property, select the property from the Properties list, then click in the Value column next to the property name.

   Depending on the property, clicking in the Value column may enable an entry field, reveal a drop list, or enable a push button used to invoke a configuration interface. A push button marked with three dots ("...") is used to indicate that you can launch an additional interface to help you make your selection.

Note that changes to the default properties in the Tools/Options interface do not affect existing screens or controls, whose properties have already been established and generated into code. But any new screen or control that is created will take on the default characteristics that you specify.

To determine which of the available screen and control properties appears in the Screen Designer Property window, do the following:

1. Open the Tools/Options interface, expand the Screen Designer tree, and select the **Visibility** category.

   This interface lists all available control properties (rather than just properties for a single control type) in alphabetical order.

2. By default, all properties in the list are displayed in the Property window. To remove a property from the list, clear the check box next to the property description, or select the property and click **Clear**.

   To add a property to the list that had previously been removed, mark the corresponding check box or select the property description and click **Set**.

3. In addition to adding or removing properties from the list one at a time, you can add or remove an entire category of properties. The check boxes that appear directly under the Set and Clear buttons list the available property categories. When you clear one of these check boxes, the check boxes corresponding to all of the properties in that category are likewise cleared. When you mark one of these check boxes, all of the properties in that category are likewise marked.

The changes that you make in this interface are saved in your personal "AcuBench80.ini" file when you click OK. If you want to save your changes to another file, you must click **Save**. To abort all changes, click **Cancel**.

13.4 Adding Screen Templates

When you create a new screen in AcuBench, you have the option to choose between several default templates (such as the blank graphical screen and blank character screen). In addition to these defaults, you have the option to create your own screen templates. Using custom screen templates can help you maintain a consistent look and feel across an application or application suite, and is an important part of consistency management.

The process of creating a screen template, described in **Chapter 14**, is very simple. Once you have the template file (".stf"), you can use the Tools/Options interface to add the template to the list that appears in the New Screen dialog.

To add a template to the New Screen dialog:

1. Open the Tools/Options interface, expand the Environment tree, and select **Template**.

   More information about the Tools/Options/Environment/Template dialog box can be found in **section 4.3.2, "Template Options."**

2. In the "Customize template" section of the interface, select **Screen** from the "Template for" drop-down box.

3. Click **Add** to open the Add New Template File dialog.

4. Enter a short, descriptive title in the "Template title" entry field, then click the browse (...) button next to the "Template file" entry field.

5. Navigate to the directory containing your ".stf", select the file, and click **Open**.

6. Add a more verbose description of the template file in the Description field.

7. Click **OK** to save your changes, then click **OK** again to close the Tools/Options interface.

   The next time you create a new screen, your template appears in the New Screen dialog, along with your description.

Tip: When you add templates to the New Screen interface, a pointer is placed in the INI file, used to locate the ".stf" file on disk. If you move the STF, AcuBench will no longer be able to locate the template and you will receive an error message. This means that if you are sharing a template among a team of developers, the STF should reside in a shared folder or be made part of your version control project.

13.5 Configuring Keyboard Shortcuts

As in the Code Editor, you can specify a custom set of keyboard shortcuts for use in the Screen Designer environment. These keyboard shortcuts can be used to perform basic editing operations (cut, copy, paste), move controls on the screen, and shape and align controls.

To add or change a keyboard shortcut associated with a Screen Designer function:

1. Open the Tools/Options menu, expand the Environment tree, and select **Keyboard**.
2. Open the Category drop-down list and select **Screen Designer**.

   The list of commands shows only those functions specific to the Screen Designer, as well as the keyboard shortcut (if any) currently assigned to each function.

   As with the Code Editor shortcuts, some common functions (like Copy or Paste) appear to have no associated keyboard shortcut. These functions have been assigned a shortcut at the "Main" level, which applies unless overridden by a shortcut assigned in a specific context.
3. Select a function in the list, then click in the Shortcut key entry field.

   If a keyboard shortcut is currently assigned to the selected function, this shortcut appears in the field. In most cases, a description of the function appears under the "Assign" and "Remove" push buttons.

4. On the keyboard, type the combination of keys that you want to assign to the selected function. The key combination appears in the entry field.

   Note the “Currently assigned to” field under the entry field. If you enter a keystroke that has already been assigned to another function, this field displays the name of that function.

5. To assign the keystroke that you have specified to the function that you have selected, click **Assign**. The new keyboard shortcut appears next to the function name.

To remove a keyboard shortcut that has already been assigned to a function, select the function in the Command list, then click **Remove**.

14 Working with Screens

Key Topics

Introduction 14-2
Creating a New Screen 14-3
Getting Started with Screen Design 14-5
Adding Controls to a Screen 14-9
Configuring Control Properties 14-25
Positioning and Aligning Controls 14-29
Refining Your Screen 14-31
Associating Code with Screen Elements 14-34
Working with Screen Templates 14-37
Generating a Screen 14-38
Testing Screens 14-39
Creating Portable Screens with AcuXUI 14-40
Creating a Logo Screen 14-41
Creating Dual User Interfaces 14-42

14.1 Introduction

Once you have created your data layout files and data sets and configured the Screen Designer environment, you are ready to start building a graphical user interface. To facilitate this process, the Screen Designer offers a WYSIWYG, drag-and-drop environment for creating character-based and graphical screens. In the designer, you can:

- Draw, position, and manipulate screen elements
- Configure screen and control properties and styles
- Associate code with screens and screen items

When you add a new screen to your program, AcuBench generates the Procedure Division code necessary to display and accept that screen. As you drag and drop controls onto the screen form, AcuBench builds a Screen Section description of the screen. Each change that you make, laying out the screen and defining interface-handling code, is recorded in the program structure file (PSF) and used to generate the COBOL code to drive your program's user interface. In other words, as you work in the Screen Designer, you are doing more than describing the appearance of a screen. You are building a fully functional user interface to your application.

This chapter discusses the process of creating new screens, designing screen layout, changing screen and control properties, associating code with screen elements, and generating screen handling code.

Note that this chapter does not discuss methods for importing existing character-based or graphical screens into the Screen Designer. These methods are discussed in **Appendix 24, "Appendix A: Bringing Existing Code Into AcuBench."**

14.2 Creating a New Screen

When you create a new project using the Standard Project template, a program containing a blank, graphical screen is automatically created within the project. Similarly, if you create a new AcuBench program within your project using the Standard Program template, the program includes a single blank, graphical screen.

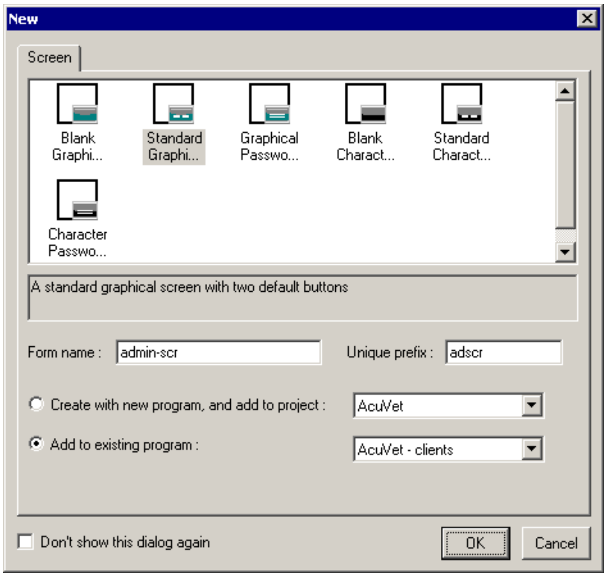
If you have created an AcuBench program that does not include a screen, or if you want to add additional screens to a program, you can use either of two methods: select a template from the New Screen interface, or open a screen template file that you have previously created. This section discusses the first method. The second method is discussed in **section 14.9**.

To add a new screen to your AcuBench program using the New Screen interface:

1. In the Workspace window's Structure view, expand the node (tree) associated with the program to which you want to add a screen.

2. Right-click the Screen folder and select **New Screen**.

   The New Screen dialog appears, listing the available screen templates.



A standard graphical screen with two default buttons

Form name : admin-scr

Unique prefix : adscr

( ) Create with new program, and add to project : AcuVet

(●) Add to existing program : AcuVet - clients

☐ Don't show this dialog again

3. Select a screen template. By default, seven templates appear on this screen. You can also use the Tools/Options interface to add additional, custom templates to this interface (see **section 13.4**).

   The default templates include the following:

Template	**Description**
Blank Graphical	A blank graphical screen
Standard Graphical	A graphical screen that contains an OK button, a Cancel button, and a main menu
Graphical Password	A graphical screen with entry fields and push buttons for collecting user login data
Blank Character	A blank character-based screen
Standard Character	A character-based screen with an OK button, a Cancel button, and a main menu
Character Password	A character-based screen for collecting user login data

4. Fill in the **Form Name** entry field with a descriptive name for your screen. Adding a descriptive name can help to minimize confusion and simplify maintenance, especially in programs with multiple screens.

   It is also possible to change the screen name after the screen has been added to your program. Just enter a new value in the **(Name)** field of the screen's property sheet.

   Note that when you change the name of a screen after the screen has been created, the prefix appended to the name of any existing controls is updated to reflect the change. The new prefix is automatically set to the first 10 characters of the screen name. See the next step for more information about prefixes.

5. The "Unique Prefix" entry field is filled in automatically with the first ten characters of your screen name, but you can change the default value. The Screen Designer uses this prefix for its default control and variable naming scheme.

   As with the screen name, you can change the screen prefix after the screen has been created. To do this, right-click the screen form in the Screen Designer and select **Change Prefix**.

6. By default, the screen will be added to the current program when you click OK. If desired, you can use the radio buttons near the bottom of the New Screen window to add the screen to a different program.
7. When you are finished making changes, click **OK**.

   The new screen form is opened in the Screen Designer, and an icon for the new screen is added to the program's Screen node in the Workspace window's Structure view. When needed, you can double-click the Structure view icon to open the screen in the Screen Designer.

14.3 Getting Started with Screen Design

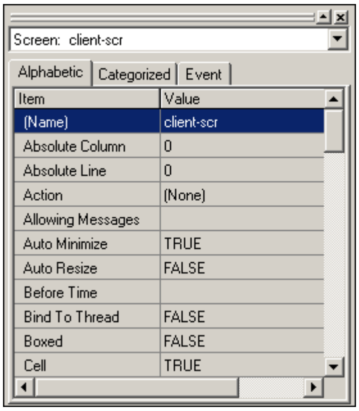
When you open a screen in the Screen Designer, the design interface comprises three main windows: the design window, the Property window, and the Screen Component Toolbox.

- The *Design window* displays a screen form. The screen form is a blank slate on which you can draw controls to build a user interface.
- The *Property window* lists the properties, events, and code insertion points associated with the selected screen element. Properties range from the basic, such as size and position of the selected item, to the advanced, dictating the style and behavior of the item.
- The *Screen Component Toolbox* lists all of the available controls (such as entry fields and grids) and other screen items (including menus and toolbars) that can be added to the selected screen form. If you are working with a character-based or AcuXUI screen, the toolbox shows a subset of the controls available for use with a graphical screen.

The first step in creating your user interface is to configure the screen itself. While screen properties can be set after controls have been added to the screen, some screen properties, such as Font, are inherited by controls added to the screen. By setting these properties at the start, you can save time and streamline the design process.

14.3.1 Setting Basic Screen Form Properties

Screen properties are configured in the Screen Designer Property window. The Property window comprises a drop-down box, which lists all of the elements that make up the current screen, and three tabs. The first two tabs, labeled Alphabetic and Categorized, list the available ACUCOBOL-GT properties for the screen element that appears in the drop-down list. The third tab, Event, lists all of the events, exceptions, and code insertion points associated with the selected screen element.



Item	Value
(Name)	client-scr
Absolute Column	0
Absolute Line	0
Action	(None)
Allowing Messages	
Auto Minimize	TRUE
Auto Resize	FALSE
Before Time	
Bind To Thread	FALSE
Boxed	FALSE
Cell	TRUE

If you have the Screen Designer open and do not see the Property window, open the View menu and select **Property Window**.

To configure screen properties:

1. Make sure that the screen form is selected in the design window.

   The screen name should appear in the drop-down box at the top of the Property window.

2. Select the Alphabetic tab of the Property window.

   As you become comfortable working in the Screen Designer, you may find it easier to work in the Categorized tab. Both tabs list all of the same properties; only the order is different. For ease of navigation, these preliminary steps use the alphabetical ordering.

3. Scroll to the top of the Property list and verify that the control's AcuBench **(Name)** property is an accurate description of the screen. The (Name) property acts as a screen handle, and is used to refer to the screen throughout the generated COBOL code.

   If you want to change the screen name, click in the Value column and type in a new name. The screen name listed in the Structure view is automatically updated when you tab or click out of the field.

   Changing the screen name also changes the unique prefix associated with the screen. Any existing control whose name includes the unique prefix is updated to indicate the new unique prefix. This prefix is the first 10 characters of the new screen name. You can change the unique prefix at any time by right-clicking the screen form and selecting **Change Prefix**.

4. By default, screen controls use the font setting defined at the screen level. It is therefore a good idea to set the screen font setting before adding controls. To change the screen font, scroll to the **Font** property and click in the Value column. You can either select one of the default fonts from the drop-down list or click the browse ("...") button to open a Font dialog that lists all available system fonts.

   The default screen font is **Small Fonts**, which corresponds to the default Windows XP system font.

5. The **Title** property is used to set the text that appears on the screen's title bar at run time. To set the screen title, scroll to the Title property, click in the Value column, and enter a literal value. When you click or tab away from the field, the title bar on the screen form is updated to reflect your changes.

6. Continue making changes as desired. Properties that you may want to configure include:

   - **System Menu**, used to determine whether or not a default system menu appears when you click the small title bar icon in the top, left corner of the screen

   - **Title Bar**, used to indicate whether or not a title will include a title bar. A window without a title bar cannot be moved by the user.

- **Window Type**, used to specify which of the four basic window types (Standard, Initial, Independent, Floating) to use for this window. A program can have only one Standard or Initial window.

For more information about windows, screens, and their properties, consult Book 2 of the ACUCOBOL-GT documentation set. In addition, Book 3, which details the syntax of the DISPLAY INITIAL WINDOW and DISPLAY FLOATING WINDOW statements, provides a comprehensive list of window properties.

14.3.2 Creating a Resizable Screen

If you plan to allow your users to resize screens at run time, you can simplify the process of handling that screen with a *layout manager*. A layout manager is a tool that, given a certain set of rules, determines how the size and placement of screen controls change as the size of the screen shifts. The exact rules vary by layout manager; you can write your own or use the one provided as part of your ACUCOBOL-GT installation, called LM_RESIZE.

This section describes how to use built-in AcuBench support for LM_RESIZE to create a resizable screen. For information about layout managers in general, as well as detailed information about the ACUCOBOL-GT resize layout manager, see section 4.8, "Layout Managers," in *ACUCOBOL-GT User Interface Programming*.

To create a resizable screen that uses LM_RESIZE as its layout manager, do the following:

1. Select the screen form in the Screen Designer.
2. In the Alphabetic tab of the Property window, scroll to the **Auto Resize** property. This property must be disabled in order to use a layout manager, so verify that its value is **FALSE**.

   If Auto Resize has been set to TRUE, click in the Value column and use the drop-down list to change its value.
3. Still in the Property window, scroll to the **Resizable** property and set its value to **TRUE**.

4. Finally, to specify a layout manager, scroll up to the Layout Manager property and change its value to **LM_RESIZE**.

When you generate your program, AcuBench automatically creates a COPY file called "lmresize.def" to support layout manager functions within your program.

Note that AcuBench also assigns a unique Layout Manager Handle property to the screen. You can change the name of the handle, but keep in mind that if you have multiple resizable screens in your application, each layout manager handle must be unique.

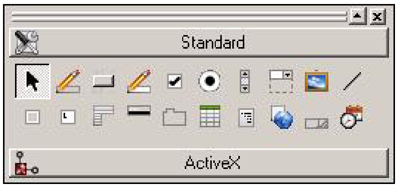
Later, as you add controls to your screen, you can use the Layout Data property to determine exactly how the resize manager handles that control when the screen size changes. The Layout Data property is discussed in **section 14.5.2**, later in this chapter.

Note that in order for the layout manager to function properly, you must also change the screen's Auto Resize property to FALSE.

The resize layout manager facility is demonstrated in an AcuBench sample project located in the Support area of the Micro Focus Web site.

14.4 Adding Controls to a Screen

Once you have set the basic properties for your screen, the next step is to add functionality by placing *screen controls* on your screen. A control is a screen element used to provide information to the user and/or receive input from the user. In the Screen Designer, you add controls to a screen form by selecting them from the Screen Component Toolbox and drawing them on the screen.



Control icons may be viewed in a list with labels or as bitmap push buttons with pop-up hints. To toggle between these views, right-click in the Screen Component Toolbox and select **List View**.

In addition to the Selector, a Menu Designer, and a Toolbar Designer, the toolbox contains the following standard controls for graphical screens:

Bar	Frame	Scroll bar
Bitmap	Grid	Status bar
Check box	Label	Tab
Combo box	List box	Tree view
Date entry	Push button	Web browser
Entry field	Radio button	

Character-based screens cannot include the bitmap, date entry, grid, status bar, tab, or Web browser controls. When you work in a character screen, the Screen Component Toolbox icons for these controls are disabled.

The Screen Component Toolbox also has a section for the ActiveX controls that you want to use in your screens. ActiveX controls are available only for graphical screens. More information about the addition of ActiveX controls to your Screen Component Toolbox can be found in **Chapter 15, section 15.3, "ActiveX Controls."**

14.4.1 Drawing Controls with the Component Toolbox

The Screen Designer lets you draw standard and ActiveX controls directly on a graphical or character-based screen form as follows:

1. With a screen form open in the Screen Designer, click a control icon in the Screen Component Toolbox.

   The icon is highlighted to indicate that it has been selected. You can change the selected control type by clicking another icon in the toolbox.

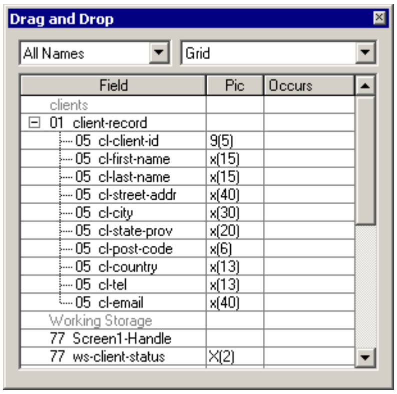
2. Move the pointer to the screen form. The pointer changes from an arrow to a crosshair.

3. Position your pointer on the screen form, then hold down the left mouse button and drag the outline that appears to size your control.

   If you release the mouse button before you have finished sizing your control, click and drag any of the small, dark blue squares that appears in the frame surrounding the control. When a control is selected, you can also use the **Shift+Arrow** keyboard shortcut to resize it one pixel at a time, or **Ctrl+Shift+Arrow** to resize it one cell at a time. (The default screen cell size is 10 pixels by 10 pixels.)

14.4.2 Drawing Controls with Drag-and-Drop

If you have defined one or more data sets within your program (as discussed in **Chapter 10, "Chapter 10: Working with Data at the Program Level"**), the data items from the corresponding FD are listed in the Screen Designer Drag and Drop pop-up window. This window also lists Working-Storage, Linkage, and other data items defined in your program.



Field	Pic	Occurs
clients		
01 client-record		
05 cl-client-id	9(5)	
05 cl-first-name	x(15)	
05 cl-last-name	x(15)	
05 cl-street-addr	x(40)	
05 cl-city	x(30)	
05 cl-state-prov	x(20)	
05 cl-post-code	x(6)	
05 cl-country	x(13)	
05 cl-tel	x(13)	
05 cl-email	x(40)	
Working Storage		
77 Screen1-Handle		
77 ws-client-status	X(2)	

By default, this interface lists the field names as they are declared in the program. If, however, you have used the XFD tab of the File Designer to assign a Name directive to a field, you can elect to have that name appear in the Drag and Drop interface instead of the field name. When you choose this option, the Title property for any control that has a Name directive will show that name, rather than the actual field name. For information about enabling this option, see **Chapter 8, section 8.4.5, "Designing a Custom XFD."**

The Drag and Drop interface allows you to select a control type and one or more data items, then draw the control on the screen. AcuBench automatically adds the data item to the control's Property sheet as a Value Variable, generated Screen Section code to assign the item as the control's VALUE property. In addition, if you select a combo box, grid, list box, paged grid, or paged list box control type, additional code is generated to populate the control with data when the screen is loaded. These controls, referred to as *autoload controls*, are discussed in **section 14.4.3**.

To use the Drag and Drop window to add controls to your screen:

1. Right-click on the screen form and select **Drag-and-Drop** from the pop-up window. You can also select the Drag-and-Drop command from the View menu.

2. From the drop-down list in the top, left portion of the interface, select a data source category. By default, "All Names" is selected, indicating that all Working-Storage, Linkage, and FD items defined in the Structure view appear in the list.

3. From the drop-down list in the top, right portion of the screen, select a control type.

4. Select one or more fields from the list of data items. To select non-contiguous items, hold down the Ctrl key as you click.

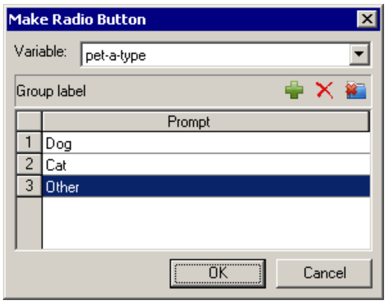
5. When you have selected all of the items that you want to draw on the screen, lift up the mouse button, position the pointer over any of the highlighted fields, then hold down the mouse button while dragging the pointer to the screen form.

   In most cases, when you release the mouse button, the control(s) is/are drawn on the screen. If you have chosen either a radio button control or one of the autoload control types, a second interface appears, allowing you to configure the behavior of the selected control. In these circumstances, when you click OK in the secondary interface, the control is drawn on the screen.

Using Drag-and-Drop to create radio buttons

Radio button controls are used when the user must select one and only one of a series of options. As a result, each individual radio button control is placed within a group, and the runtime ensures that only one member of the group at a time can be selected.

When you use Drag-and-Drop to create a radio button, the initial steps are the same as any other control. You select a control type and a data item, then drag the data item to the screen. When you do this, the Make Radio Button dialog appears.



Variable: pet-a-type

	Prompt
1	Dog
2	Cat
3	Other

This interface represents a group of radio buttons. You add individual radio button controls to the group corresponding to each of the available options.

To add items to the group of radio buttons:

1. By default, the name of the selected data item appears as the first and only radio button in the group. To change the name of the first radio button to something more descriptive, double-click the field name in the Prompt list and add a more descriptive name.

   The name listed here is assigned as the control's title, which appears next to the radio button on the screen.

2. Click the **Add** button. The Add button is the first of the three buttons just above the Prompt list, on the right side of the interface.

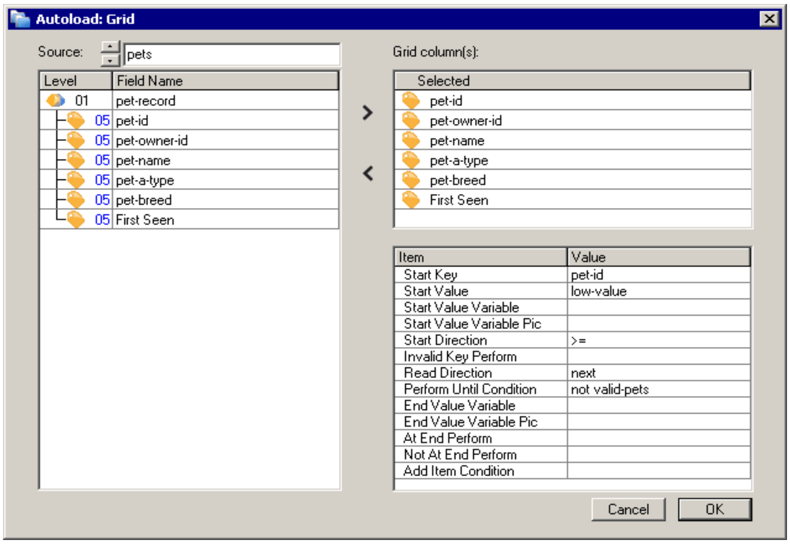
3. The new radio button is assigned the default name "Radio1". As with the first radio button, double-click in the Prompt field to enter a more descriptive name for the button.

4. Repeat this process until you have added all necessary group items.

5. When you are finished, click **OK**. The group of radio buttons that you have described is drawn on the screen. You can now size and position the controls on the screen.

14.4.3 Creating Autoload Controls

As with radio buttons, when you use the Drag and Drop window to create a combo box, grid, or list box, a secondary interface opens. This interface is called the Autoload window.



Level	Field Name
01	pet-record
05	pet-id
05	pet-owner-id
05	pet-name
05	pet-a-type
05	pet-breed
05	First Seen

Selected
pet-id
pet-owner-id
pet-name
pet-a-type
pet-breed
First Seen

Item	Value
Start Key	pet-id
Start Value	low-value
Start Value Variable	
Start Value Variable Pic	
Start Direction	>=
Invalid Key Perform	
Read Direction	next
Perform Until Condition	not valid-pets
End Value Variable	
End Value Variable Pic	
At End Perform	
Not At End Perform	
Add Item Condition	

The Autoload window

The Autoload window lets you add or remove data items associated with the control and determine how data items are loaded into the control and displayed at runtime. It is important to note that, unlike other controls created through the Drag-and-Drop interface, autoload controls can be associated only with data items from a data set (defined in an FD). This means that you can *not* use the Autoload feature to load a control with data from a table in Working-Storage. In addition, each Autoload control can take its data from only a *single* data set (FD). Autoload controls can only be associated with data sets that are associated with indexed files.

Once an autoload control has been generated, any additional controls created by the autoload (like labels) cannot be changed, because that part of the code will not be regenerated.

If you use the Drag-and-Drop function to create a combo box, grid, or list box and select items from multiple data sets or from non-FD data items, the Autoload interface is not displayed and the generated code associated with autoload controls is not created. A control is drawn on the screen, but the code to load the control is not generated. You must manually add the code needed to load the control.

When you create an autoload control, the list in the bottom, right portion of the Autoload window allows you to customize the code generated to load the control. If you want to read data into a list box control from an indexed file, for example, you can specify a key, a start value, and a read direction, as well as a code to use when certain conditions (AT END, ADD ITEM) are encountered.

When you have finished defining the control data in the Autoload window and click **OK**, AcuBench draws the control on the screen form. You can then position, resize, or align the control just like any other screen control.

You can create five types of controls using the Autoload function: combo box, grid, list box, paged grid, and paged list box. Note that because the code used to load and navigate through a paged control is quite different from that used with a typical grid or list box, the Autoload function requires you to distinguish between the paged and unpaged styles when the control is created. This means that if you use the control's property sheet to change its STYLE property after autoload code has been created, data will no longer automatically appear in the control, and you may receive compiler errors.

14.4.3.1 Creating an unpaged autoload control

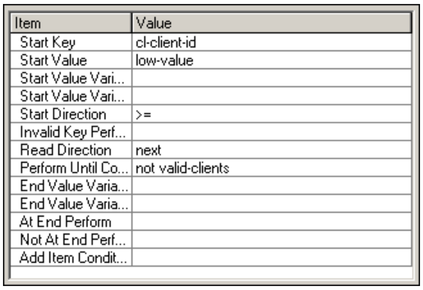
When you create an unpaged control—a combo box, list box, or grid—using the Autoload interface, the list in the bottom, right portion of the window provides a number of ways for you to modify the generated code used to load data into the control at runtime.

The steps are as follows:

1. Open the Drag and Drop window and select **Combo-Box**, **Grid**, or **List-Box** from the list of controls.
2. Select one or more fields from a single data set in the list of available data items. Note that combo boxes take only a single data field (loaded into a single column of data).
3. Click **OK**. This opens the Autoload interface.
4. On the left side of the screen, verify that you have chosen the intended data set and data fields.

   To change data sets, use the arrow keys next to the Source entry-field. To add data fields to the control, select the new field, then click the right arrow symbol. To remove data fields from the control, highlight the item in the Selected list and click the left arrow symbol.
5. Use the list in the bottom, right portion of the screen, shown below, to determine how the generated code will load data into the control at runtime.

   The interface is similar to that of the Screen Designer Property window:



Item	Value
Start Key	cl-client-id
Start Value	low-value
Start Value Vari...	
Start Value Vari...	
Start Direction	>=
Invalid Key Perf...	
Read Direction	next
Perform Until Co...	not valid-clients
End Value Varia...	
End Value Varia...	
At End Perform	
Not At End Perf...	
Add Item Condit...	

You can enter variable names, paragraph names, or condition clauses to be inserted into the AcuBench-generated load paragraph for the control. The options are as follows:

Option	Description
Start Key	Specifies which key to use when reading data into the control from a file.
Start Value	The value used to start reading the record. The default is **low-value**.

Option	**Description**
Start Value Variable	If you want the start value to vary depending on some condition, you can specify a start value variable. AcuBench adds this variable to Working-Storage and uses it during code generation.
Start Value Variable Pic	The picture clause for the start value variable that you have specified.
Start Direction	This determines the KEY clause of the START phrase. By default, the value is >=: `START filename KEY >= keyname`
Invalid Key Perform	Allows you to specify a paragraph to be performed when an INVALID KEY condition is encountered.
Read Direction	Determines whether a file is read start to end or end to start. The default value is **next**.
Perform Until Condition	Allows you to enter the end condition for the PERFORM statement used to load the control. Uses the same syntax as the Expression Builder. The default is **not valid-*filename***. "Valid-*filename*" is an AcuBench-generated file status condition.
End Value Variable	Used with "Perform Until Condition" to specify when to stop loading information into the control. Enter a variable to be used to determine the end condition for the PERFORM statement used to load the control. AcuBench adds this variable to Working-Storage.
End Value Variable Pic	The picture clause for the end value variable.
At End Perform	Allows you to specify a paragraph to be performed when an AT END condition is encountered.
Not At End Perform	Allows you to specify a paragraph to be performed when a NOT AT END condition is encountered.

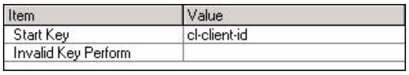
Option	Description
Add Item Condition	Create a condition to determine whether or not to add a record to the control. Uses the same syntax as the Expression Builder.

6. When you have finished making changes, click **OK**. AcuBench draws the control on the screen form.

The code generated to load unpaged controls is generated into the ".prd" COPY file and should not be modified directly.

14.4.3.2 Creating a paged autoload control

Much of the code for paged controls is generated into the Event Paragraph and made available for direct editing. This code is created once, then left untouched to preserve any modifications that you might make. Because you can customize the code directly, rather than going through a graphical interface, the list in the bottom, right portion of the Autoload window is very simple for paged controls. It contains the only two options that affect the portion of the autoload code generated into the write-protected ".prd" file.



Item	Value
Start Key	cl-client-id
Invalid Key Perform	

The initial steps used to create paged autoload controls are similar to the steps used to create unpaged controls, with the following differences:

1. In the Drag-and-Drop window, choose **Paged Grid** or **Paged List-Box** as the control type.

2. In the Autoload window, you can modify the Start Key selection or add an Invalid Key Perform Paragraph. The other options listed for unpaged controls do not appear.

3. When you are finished, click **OK**. AcuBench draws the control on the screen form and immediately adds code to the Event Paragraph to control how data is loaded into the control. When you generate your program, a very small amount of additional code is generated in the ".prd" COPY file to manage some portions of the autoload process for the paged control. The code in the ".prd" file should not be modified directly.

The code added to the Event Paragraph when the control is drawn on the screen is appended to whatever event code you have already created. To allow you to make free use of this autoload code, AcuBench does not modify or regenerate the code once it has been created. This means that any changes that you make will be preserved when the program is generated. Because AcuBench does not retain control of this code once it has been added to the Event Paragraph, if you delete the control with which the code is associated, you must manually delete the code.

Please note that code added to the Event Paragraph for your use is **not** deleted if you delete the paged control from the screen form. Like event code that you create, the AcuBench-created autoload code must be manually deleted from the Event Paragraph when it is no longer being used.

As a corollary to this, Working-Storage items associated with a paged autoload control must also be manually deleted. Because you are free to make changes to these variables to support modifications to the code in the Event Paragraph, AcuBench does not retain control over these variables and will not delete them.

Caution: If you delete a paged autoload control, but do not delete the associated code in the Event Paragraph, you will receive compiler errors. Also, you must delete any Working-Storage items associated with the deleted control before creating another paged autoload control with the same name. If you do not, undefined behavior is likely to result.

14.4.3.3 Modifying autoload controls

Although you cannot change an autoload control's style from PAGED to UNPAGED (or vice-versa), you can customize other aspects of the control's appearance or behavior using the Property window, just as for other controls. You can change the control font, background colors, column sizes, overall height and width, layout data information, event and exception procedures, and so on, without affecting the automatically-generated code used to load the control at runtime.

To make it easier to modify the code generation for controls created using the Autoload window, a special property has been added to the Property window for combo box, grid, and list box controls. This property, Autoload, indicates whether a control has been created using the Autoload feature. If Autoload is set to “1”, AcuBench is generating the code to load the control. If Autoload is set to “0”, the control is being loaded manually. This special property is used only by the Screen Designer and does not appear in the Screen Section definition for the control. If you have created a control manually, you cannot set its Autoload property to “1” to prompt AcuBench to start generating the code for the control. Instead, open Drag-and-Drop and create a new control using the Autoload feature.

To modify the data portion of a control created through the Autoload interface:

1. Select the control in the Screen Designer.
2. In the property window, next to the **Autoload** property, click in the Value column.
3. Click the **Browse** (...) button to open the Autoload window for the selected control.

   You can now change which fields are displayed in the control, which direction the data file is read, or even which FD is used to read records into the control.

Note that if you make changes to a control’s layout that are tied to the data in the control (modifying the text of the heading cells in a grid or changing column widths in a grid or list box), those changes will be lost if you open the Autoload dialog and change your field selections. Other sorts of property changes, such as selecting a new background color or adjusting the font, are not affected by modifications made through the Autoload interface. Changes cannot be made to additional controls created by the Autoload interface, such as labels.

14.4.3.4 Understanding the generated code (unpaged controls)

Although the code generated for each control type is different, based on the unique characteristics of the control, the code generated for each of the unpaged controls is very similar. The first example shows the code generated into the Procedure Division (“.prd”) COPY file for a grid control that takes its data from a data file called “clients”. The second example shows the code skeleton from which the first sample was generated.

Generated code sample: Grid

```
MOVE LOW-VALUE TO cl-client-id.
START clients, KEY >= cl-client-id
   INVALID KEY
      PERFORM Invalid-Key-Paragraph
END-START.
PERFORM UNTIL NOT valid-clients
   READ clients NEXT RECORD
      AT END
         PERFORM At-End-Paragraph
      NOT AT END
         IF 2>1
            PERFORM Not-At-End-Paragraph
         END-IF
   END-READ
END-PERFORM.
MODIFY clients-gd, MASS-UPDATE = 0.
```

Generated code skeleton: Grid

```
MOVE [Start Value / Variable] TO [Start Key].
START [Filename], KEY [Start Direction] [Start Key]
   INVALID KEY
      PERFORM [Invalid Key Perform]
END-START.
PERFORM UNTIL [Perform Until Condition] [End Value / Variable]
   READ [Filename] [Read Direction]
      AT END
         PERFORM [At End Perform]
      NOT AT END
         IF [At Item Condition]
            PERFORM [Not At End Perform]
         END-IF
   END-READ
END-PERFORM.
MODIFY [Control-Name], MASS-UPDATE = 0.
```

14.4.3.5 Understanding the generated code (paged controls)

The code used to load data into a paged control differs significantly from that used to load data into an unpaged control. In the latter case, all data from the specified source is loaded into the control (and into memory) when the control is displayed on the screen. With a paged control, however, only the data visible in the control (that is, a single page of data) is loaded into memory. This can result in a dramatic performance improvement when the data source contains large numbers of records.

Note that the method used to load data into a paged grid is substantially different from the method used to load data into a paged list box, because of fundamental differences in the control type.

Generated Event Paragraph code sample: Paged Grid

```
* Autoload paged control code generated by Drag-And-Drop
 Screen1-Gd-1-Ev-Msg-Paged-Next.
     PERFORM event-data-2 TIMES
        READ clients NEXT RECORD
           AT END
              MOVE event-action-fail TO event-action
              EXIT PARAGRAPH
        END-READ
     END-PERFORM.
     PERFORM ACU-Screen1-Gd-1-Autoload-Add.
* Autoload paged control code generated by Drag-And-Drop
 Screen1-Gd-1-Ev-Msg-Paged-Prev.
     PERFORM event-data-2 TIMES
        READ clients PREVIOUS RECORD
           AT END
              MOVE event-action-fail TO event-action
              EXIT PARAGRAPH
        END-READ
     END-PERFORM.
     PERFORM ACU-Screen1-Gd-1-Autoload-Load.
     MODIFY Screen1-Gd-1, INSERTION-INDEX=2,
     RECORD-TO-ADD = Screen1-Gd-1-Autoload.
* Autoload paged control code generated by Drag-And-Drop
 Screen1-Gd-1-Ev-Msg-Paged-First.
     MOVE LOW-VALUES TO cl-client-id
     START clients, KEY >= cl-client-id
        INVALID KEY
           MOVE event-action-fail TO event-action
     END-START.
* Autoload paged control code generated by Drag-And-Drop
 Screen1-Gd-1-Ev-Msg-Paged-Last.
     MOVE HIGH-VALUES TO cl-client-id
     START clients, KEY <= cl-client-id
        INVALID KEY
           MOVE event-action-fail TO event-action
    END-START.
```

Generated Event Paragraph code sample: Paged List-Box

```
* Autoload paged control code generated by Drag-And-Drop
 Screen1-Lb-1-Ev-Ntf-Pl-Next.
     PERFORM Screen1-Lb-1-get-next-item .
* Autoload paged control code generated by Drag-And-Drop
 Screen1-Lb-1-Ev-Ntf-Pl-Prev.
     PERFORM Screen1-Lb-1-get-prev-item .
* Autoload paged control code generated by Drag-And-Drop
 Screen1-Lb-1-Ev-Ntf-Pl-Nextpage.
     MODIFY Screen1-Lb-1, MASS-UPDATE = 1
     PERFORM Screen1-Lb-1-Get-Next-Item
             Screen1-Lb-1-page-size times
     MODIFY Screen1-Lb-1, MASS-UPDATE = 0.
* Autoload paged control code generated by Drag-And-Drop
 Screen1-Lb-1-Ev-Ntf-Pl-Prevpage.
     MODIFY Screen1-Lb-1, MASS-UPDATE = 1
     PERFORM Screen1-Lb-1-Get-Prev-Item
             Screen1-Lb-1-page-size times
     MODIFY Screen1-Lb-1, MASS-UPDATE = 0.
* Autoload paged control code generated by Drag-And-Drop
 Screen1-Lb-1-Ev-Ntf-Pl-First.
     MOVE LOW-VALUES TO cl-client-id
     START clients, KEY NOT < cl-client-id
     END-START.
     SET Screen1-Lb-1-READING-FORWARDS TO TRUE
     MODIFY Screen1-Lb-1, MASS-UPDATE = 1
                          RESET-LIST = 1.
     PERFORM Screen1-Lb-1-get-next-item
             Screen1-Lb-1-page-size times
     MODIFY Screen1-Lb-1, MASS-UPDATE = 0.
* Autoload paged control code generated by Drag-And-Drop
 Screen1-Lb-1-Ev-Ntf-Pl-Last.
     MOVE HIGH-VALUES TO cl-client-id
     START clients, KEY NOT > cl-client-id
     END-START.
     SET Screen1-Lb-1-READING-BACKWARDS TO TRUE
     MODIFY Screen1-Lb-1, MASS-UPDATE = 1
                          RESET-LIST = 1.
     PERFORM Screen1-Lb-1-get-prev-item
             Screen1-Lb-1-page-size times
     MODIFY Screen1-Lb-1, MASS-UPDATE = 0.
* Autoload paged control code generated by Drag-And-Drop
 Screen1-Lb-1-get-next-item.
     EVALUATE TRUE
```

```
         WHEN Screen1-Lb-1-at-start
             MOVE low-value to cl-client-id
             START clients, key not < cl-client-id
             END-START
             ADD 1 to Screen1-Lb-1-page-size
                 GIVING Screen1-Lb-1-number-reads-needed
         WHEN Screen1-Lb-1-at-end
              EXIT paragraph
         WHEN Screen1-Lb-1-reading-backwards
              MOVE Screen1-Lb-1-page-size to
                   Screen1-Lb-1-number-reads-needed
         WHEN Screen1-Lb-1-reading-forwards
              MOVE 1 to Screen1-Lb-1-number-reads-needed
     END-EVALUATE.
     PERFORM Screen1-Lb-1-number-reads-needed times
        READ clients next record
             AT END
                SET Screen1-Lb-1-at-end TO TRUE
                EXIT PARAGRAPH
        END-READ
     END-PERFORM.
     INITIALIZE Screen1-Lb-1-Autoload
     PERFORM Acu-Screen1-Lb-1-Autoload-Load
     MODIFY Screen1-Lb-1, item-to-add = Screen1-Lb-1-Autoload
     SET Screen1-Lb-1-reading-forwards TO TRUE.
* Autoload paged control code generated by Drag-And-Drop
 Screen1-Lb-1-get-prev-item.
     EVALUATE TRUE
        WHEN Screen1-Lb-1-at-end
           MOVE high-value to cl-client-id
           START clients, KEY NOT > cl-client-id
           END-START
           ADD 1 TO Screen1-Lb-1-page-size
                 GIVING Screen1-Lb-1-number-reads-needed
        WHEN Screen1-Lb-1-at-start
           EXIT paragraph
        WHEN Screen1-Lb-1-reading-forwards
           MOVE Screen1-Lb-1-page-size TO
                Screen1-Lb-1-number-reads-needed
        WHEN Screen1-Lb-1-reading-backwards
           MOVE 1 TO Screen1-Lb-1-number-reads-needed
     END-EVALUATE.
     PERFORM Screen1-Lb-1-number-reads-needed times
        READ clients previous record
           AT END
```

```
             SET Screen1-Lb-1-at-start TO TRUE
             EXIT PARAGRAPH
          END-READ
       END-PERFORM.
       INITIALIZE Screen1-Lb-1-Autoload
       PERFORM Acu-Screen1-Lb-1-Autoload-Load
       MODIFY Screen1-Lb-1, insertion-index = 1
       ITEM-TO-ADD = Screen1-Lb-1-Autoload
       SET Screen1-Lb-1-reading-backwards TO TRUE.
```

14.5 Configuring Control Properties

When you select a control in the Screen Designer, the Property window lists the control's name and all of the properties currently associated with that control. The properties listed vary according to the control type, so the properties listed for a bitmap control, for example, are going to be very different from the properties listed for an entry field.

When you select multiple controls in the Screen Designer, the Property window lists only those properties that apply to all of the selected controls. If you change one of these properties, the change is applied to every one of the selected controls.

The default Property settings for each control type, as well as the properties that appear in the Property window, can be configured through the Tools/Options interface, as discussed in **Chapter 13, section 13.3, "Establishing Screen and Control Defaults."**

Detailed information about all of the available control properties can be found in Book 2 of the ACUCOBOL-GT documentation set.

To change control property settings in AcuBench:

1. In the Screen Designer, select the control on the screen form. The control name appears in the drop-down list at the top of the property window.

   Note that if you have selected multiple controls, only the name of the last control selected is shown.

2. Use the Alphabetic or Categorized tab of the Property window to scroll through the available properties and find the one you wish to change.

3. Click in the Value column next to the property description to change the control's property settings. In some cases, this activates an entry field in which you can type the new value. In other cases, you may select the value from a drop-down list, or click a browse ("...") button to open a secondary interface used to specify a property value.

In general, the first property changed for each control is the (Name) property from the AcuBench default to something more descriptive. As with the screen itself, this (Name) property is used to assign a control name used to refer to the control programmatically. AcuBench also uses this value to construct the default names for paragraphs and variables associated with the control, so assigning a descriptive name plays an important role in keeping your code readable and intuitive. If you are tempted to skip this step and use the default names, imagine trying to code a screen full of entry fields called Screen1-Ef-1, Screen1-Ef-2, Screen1-Ef-3, and so on. Now imagine an entire application full of screens and fields using that same default naming convention.

14.5.1 Associating Data with a Control

When you use the Drag and Drop interface, controls that you draw on the screen are automatically associated with data items, usually from graphical Working-Storage or from a data set. You can also manually associate a value or variable with a screen control.

In the Screen Section, the VALUE property determines what data, if any, is associated with a given control. In the AcuBench Screen Designer, two Property window items are associated with this property. The *Value* property entry is used to set a fixed, initial value shown in the control when in the screen is displayed. The *Value Variable* property is used to tie the control to a variable data item, which may or may not contain a value when the screen is displayed. When these two properties are both set, the Value property is used to set the initial value for the specified Value Variable.

If you need to change the Exception Variable and the Exception Value, please make sure that you change the Exception Variable first and then the Exception Value, because that might cause the Exception Variable to disappear from Working Storage.

To associate a variable with a control without using the Drag and Drop interface:

1. In the Screen Designer, select a control on the screen form.
2. Scroll through the Property window until all four value properties (Value, Value Multiple, Value Picture, and Value Variable) are visible.
3. Click in the Value column of the Property window next to the **Value Variable** listing and do one of the following:
    - To create a new Working-Storage variable associated with the selected control, type a variable name in the field. When you tab or click away from the field, AcuBench automatically adds the new variable to graphical Working-Storage.
    - To select from a list of variables already assigned to other controls on the screen, expand the drop-down list and choose a value.
    - To select from a list of data items associated with your program's data sets, graphical Working-Storage, and so on, click the browse ("...") button and select an item from the interface.
4. When you make an entry for the Value Variable property, the **Value Picture** property is updated to reflect the PICTURE clause associated with that variable.

    If you have entered a new variable name, a default value appears in this field. To change the value, click in the Value column next to the Value Picture property and enter a new value. AcuBench automatically updates the variable definition in graphical Working-Storage.
5. If you want to set an initial value for the specified variable, click in the Value column for the **Value** property and enter the appropriate literal value.

If you select an entry field, for example, and assign a Value Variable property of “ws-name”, a Value Picture property of “X(30)”, and a Value property of “<Enter Name Here>”, AcuBench generates the following code:

- In graphical Working-Storage (and the “.wrk” COPY file):

```
77  ws-name   PIC X(30)  VALUE IS "<Enter Name Here>".
```

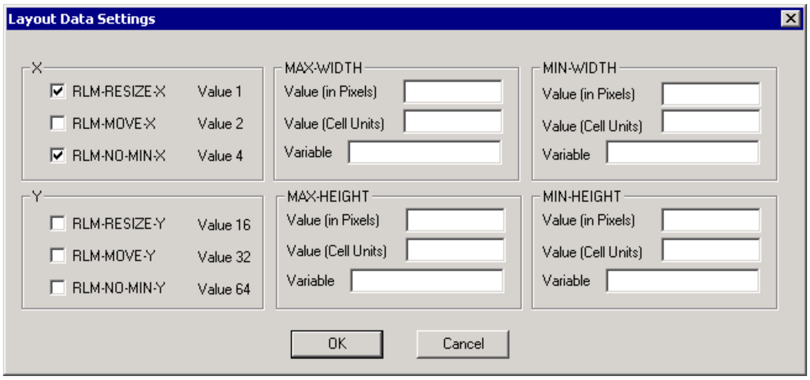
- In the “.scr” COPY file:

```
03  main-scr-name-ef, Entry-Field, ...
    VALUE ws-name.
```

14.5.2 Layout Data Control Property

As mentioned in **section 14.3.2, “Creating a Resizable Screen,”** when you want the resize layout manager to change a control’s appearance when a screen is resized, you set that control’s Layout Data property to a non-zero value. The different Layout Data property values and how they affect a control’s behavior are described in section 4.8.4.1, “Resize manager LAYOUT-DATA values,” in *ACUCOBOL-GT User Interface Programming*. When Layout Data is the default value of “0”, no code is generated.

In AcuBench, these values are set via the Layout Data Settings dialog box, which you access by clicking in the Layout Data value box in the Property window.


Layout Data Settings

X
- [✓] RLM-RESIZE-X Value 1
- [ ] RLM-MOVE-X Value 2
- [✓] RLM-NO-MIN-X Value 4

Y
- [ ] RLM-RESIZE-Y Value 16
- [ ] RLM-MOVE-Y Value 32
- [ ] RLM-NO-MIN-Y Value 64

MAX-WIDTH: Value (in Pixels), Value (Cell Units), Variable

MIN-WIDTH: Value (in Pixels), Value (Cell Units), Variable

MAX-HEIGHT: Value (in Pixels), Value (Cell Units), Variable

MIN-HEIGHT: Value (in Pixels), Value (Cell Units), Variable

Note that you can set either a resize or a move value for each axis (*x* or *y*), but you cannot set both values for any axis. Minimum and maximum dimensions for a control (height and width) are also set in this dialog. Refer to section 4.8.3, "Minimum and Maximum Control Dimensions," in *ACUCOBOL-GT User Interface Programming* for detailed information about these settings.

14.5.3 Controls: Related References

In most instances, controls are simple to create and manage. In some cases, however, they are complex. In addition to the information contained in this chapter, detailed information pertaining to Property window items is located in the following volumes and chapters in the ACUCOBOL-GT documentation set:

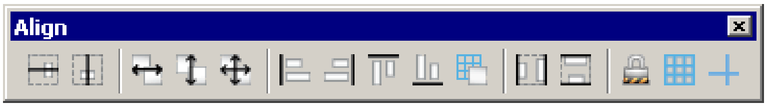
1. Book 2, Chapter 3, "Graphical Controls," provides an overview of controls. It also describes the creation and use of bitmap buttons and the general use of paged list boxes.
2. Book 2, Chapter 4, "Supporting Concepts and Related Issues," contains background material concerning issues related to the creation of controls and screens.
3. Book 2, Chapter 5, "Control Types Reference," describes individual controls in detail, including styles and properties and how they can be set. It also lists the events that can be associated with each control.
4. Book 2, Chapter 6, "Events Reference," describes in detail the events that can be generated when you use windows and controls in an event-driven environment.
5. Book 3, Chapter 5, section 5.9, "Screen Description Entry," contains syntax rules and general rules for the components of a screen description entry.
6. Book 3, Chapter 6, section 6.4.9, "Common Screen Options," describes options that are common to Screen section entries and the ACCEPT, DISPLAY, and MODIFY verbs. It also includes the phrases used to specify a control's common properties.

7. Book 3, Chapter 6, section 6.6, “Procedure Division Statements,” contains the DISPLAY *control-type* format of the DISPLAY verb. It also contains syntax and general rules for the use of this DISPLAY verb format.

A general overview of ActiveX and OLE programming issues appears in Chapter 3, section 3.4, “Using ActiveX Controls and COM Objects in Your COBOL Program,” in *A Guide to Interoperating with ACUCOBOL-GT.*

14.6 Positioning and Aligning Controls

Once you have added controls to your screen, you can use arrow-key shortcut combinations, the Align menu, or Align toolbar to fine-tune the size and position of those controls.



You can use the following keystrokes to position and resize controls:

- Use the left, right, up, or down arrow controls to move the selected control(s) one pixel at a time in the specified direction.
- Hold down the Ctrl key and use the arrow keys to move the selected control one cell at a time in the specified direction.
- Hold down the Shift key and use the arrow keys to resize the control one pixel at a time (the up and down arrows move the bottom border of the control, making the control taller or shorter; the left and right arrows move the right border of the control, making it wider or narrower).
- Hold down the Ctrl and Shift keys and use the arrow keys to resize the control one cell at a time.

When you use most of the Align commands, a group of selected controls is repositioned in relation to the last control selected, known as *reference control*. The reference control is marked by the small, dark blue boxes (handles) on the control frame. (The handles on other selected controls are white, with a blue border.)

The Align commands that adjust controls in relation to one another include:

- **Make Same Size (Width, Height, or Both)**, used to resize all of the selected controls to the width, height, or width and height of the reference control. From left to right, these are the third, forth, and fifth buttons on the Align toolbar.
- **Align Control (Left, Right, Top, or Bottom)**, used to align the selected controls to the left, right, top, or bottom border of the reference control From left to right, these are the sixth through ninth buttons on the Align toolbar.
- **Space Evenly (Across or Down)**, used to adjust controls so that a consistent amount of horizontal or vertical space appears between the selected controls. This control is only enabled when three or more controls are selected.
- **Adjacent (Horizontal or Vertical)**, used to align the selected controls so that the borders of the controls are immediately adjacent to one another in either the horizontal or vertical direction. This is useful, for example, in positioning buttons on a toolbar. These options appear just to the left of the Lock button on the Align toolbar.

The Align commands that can be used to position either a single control or multiple controls include:

- **Align Control (Center Horizontal, Center Vertical, or To Grid)**, used to center one or more controls on the screen, or to align the control(s) to the closest grid point on the screen form. Center Horizontal and Center Vertical are the first two buttons on the Align toolbar; Align To Grid appears just to the right of the Align Bottom button.
- **Lock Controls**, a toggle switch used to lock controls in place on the screen form so that they cannot be accidentally moved, or to unlock controls to allow further positioning. A locked control cannot be dragged, centered, or sized, nor can it be used as the dominant control when a group of controls is selected. The selection handles on a locked control are white instead of dark colored.

- **Size to Content**, used to adjust the borders of the control to match the content of the control. This is frequently used with controls like labels and radio buttons to match the control precisely to the size of its label, eliminating unused space. This option does not appear on the Align toolbar, but does have a keyboard shortcut: Shift+F7.

By default, when you work in the Screen Designer, a grid of dots appears on the screen form to help you in positioning your controls. The size of each square in this grid corresponds to the screen's CELL SIZE property (by default, 10 pixels by 10 pixels). You can toggle whether or not this grid is visible with the **View Grid** command, accessed through the Align menu or toolbar. To help you place items on this grid, you can enable a set of control positioning guides with the **Toggle Guide** command. These positioning guides become visible when you click on a control on the screen form, or click to draw a control on the screen.

14.7 Refining Your Screen

As you work with and refine your screen, a number of useful editing functions are available to help you as you work. Most of these functions, like Undo/Redo or Cut/Copy/Paste, can be invoked with keyboard shortcuts. You can also use the Edit menu or the Screen Designer right-click pop-up menu to access most editing commands.

Remember that most keyboard shortcuts can be customized through the Tools/Options interface, as discussed in **Chapter 13, section 13.5, "Configuring Keyboard Shortcuts."**

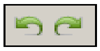
Keep in mind that you can use the Tab key to navigate through the controls in your screen. Press **Tab** to move through the controls on the screen from lowest tab order number to highest. Press **Shift+Tab** to move backward through the controls. Screen control tab order is discussed in more detail in **section 14.7.2**.

14.7.1 Basic Editing Commands

AcuBench uses the standard set of commands and keyboard shortcuts to invoke basic editing commands. These commands include the following:

- To reverse your most recent action, choose the **Edit/Undo** (Ctrl+Z) command. You can restore the action that you reversed by choosing **Redo** (Ctrl+Y). The workbench supports multiple undo and redo actions, in reverse order until the last save performed.

  You can also use the Undo and Redo buttons on the Standard toolbar.

- To cut and copy controls to the clipboard, select the control(s) that you want to cut or copy and choose the **Edit/Cut** (Ctrl+X) or **Edit/Copy** (Ctrl+C) commands. **Cut** removes the control from the screen and places it on the clipboard. **Copy** adds a duplicate of the selected control to the clipboard.

- After you cut or copy a control to the clipboard, you can paste it to a screen form (Edit/Paste or Ctrl+V). If you paste the control from one screen form to another, the control is placed in the same location on the new screen that it occupied on the original screen. After pasting a control, you can move it to any location in the screen form.

  A pasted control retains many of the original control's properties. However, it does not retain such unique properties as ID, name, and tab order. The Screen Designer assigns default unique properties, which you can change.

  After pasting multiple, identical controls, you can quickly change an individual property value so that it is unique for each control copy. Select all the control copies that you want to have unique variable names (for example, select five identical entry fields) and change the desired variable name in the Property window. AcuBench detects the multiple selection and asks if you want to give each selected control a unique name for that property. With a **Yes** response, AcuBench automatically increments the number in the variable name for each control so it is unique.

14.7.2 Determining Control Tab Order

Control tab order is determined by the order in which controls are drawn on the screen. The first control that you draw is assigned a tab order of "1", the next is given tab order number "2", and so on. These tab order numbers correspond to the order in which controls appear in the Screen Section definition for the screen.

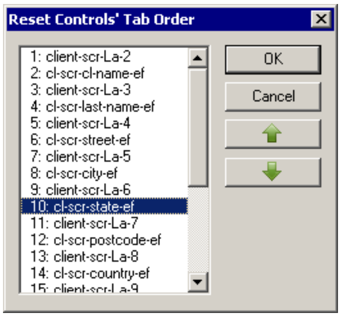
When a user presses the Tab key to move through the screen, tab order determines the default behavior for moving the cursor. Unless you have defined alternate behavior for the Tab key (or otherwise added special keystroke handling code), when the user presses Tab, the cursor moves to the next ACCEPT field on the screen.

As you cut, paste, and move controls around a screen form in the Screen Designer, you are likely to need to make adjustments to the controls' tab order. These changes can be made quickly and easily through the Reset Controls' Tab Order dialog.

To view and adjust controls' tab order, do the following:

1. Select the **Tab Order** command from the Screen Designer right-click menu or the Align menu to open the Reset Controls' Tab Order dialog. (You can also use the **Ctrl+D** keyboard shortcut.)

   The dialog box lists all of the controls on your screen in tab order, next to the tab order number.

   1: client-scr-La-2
   2: cl-scr-cl-name-ef
   3: client-scr-La-3
   4: cl-scr-last-name-ef
   5: client-scr-La-4
   6: cl-scr-street-ef
   7: client-scr-La-5
   8: cl-scr-city-ef
   9: client-scr-La-6
   10: cl-scr-state-ef
   11: client-scr-La-7
   12: cl-scr-postcode-ef
   13: client-scr-La-8
   14: cl-scr-country-ef
   15: client-scr-La-9

   The tab order numbers also display on the screen design form.

2. Change the tab order on your screen by selecting a control and using the up and down arrows.

   Note that changing the tab order also changes the order that controls appear in the Screen Section. In other words, the user tabs through controls on the screen in the order that those controls appear in the program's Screen Section definition.

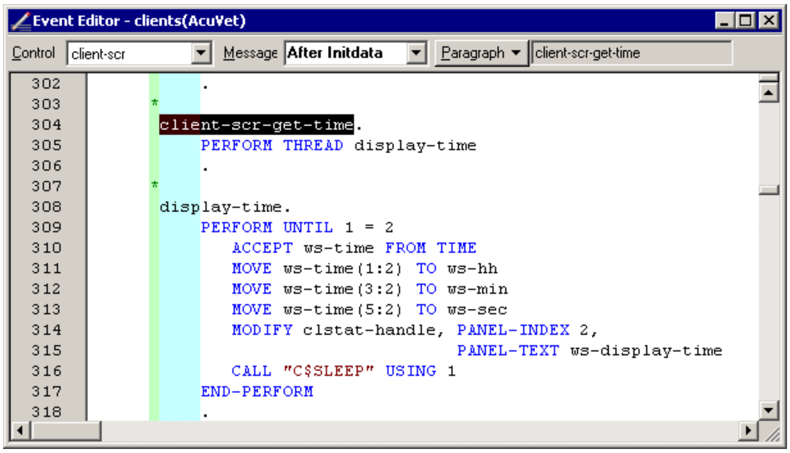
3. You can also use the **Send to Back** and **Send to Front** commands in the Screen Designer right-click pop-up menu. Send to Back gives the selected control the lowest tab order number, reordering all other controls accordingly. Send to Front gives the selected control the highest tab order number. These controls are not enabled when more than one control is selected.

Note that if you create a frame and place controls inside that frame, the tab order within the frame exists exclusively of the controls outside of it. In this instance, you cannot create a tab order outside of a frame and continue it inside of the frame. The tab order inside of the frame starts at "1." You can, however, change the tab order within the frame itself. Tab order is continuous if you draw a frame around controls that already exist.

14.8 Associating Code with Screen Elements

Once you have added controls to the screen form and made any necessary configuration changes, the next step is to add the code to make the user interface function. The Screen Designer interface used to associate code with screens and screen controls is the Event Editor.

The Event Editor interface is very similar to that of the Code Editor. The main visible difference is at the top of the Event Editor window, where three fields indicate the screen element to which the code applies, the type of code being associated with the screen element, and the name of the paragraph containing that code.



```
                .
       *
       client-scr-get-time.
           PERFORM THREAD display-time
           .
       *
       display-time.
           PERFORM UNTIL 1 = 2
              ACCEPT ws-time FROM TIME
              MOVE ws-time(1:2)  TO ws-hh
              MOVE ws-time(3:2)  TO ws-min
              MOVE ws-time(5:2)  TO ws-sec
              MODIFY clstat-handle, PANEL-INDEX 2,
                                    PANEL-TEXT ws-display-time
              CALL "C$SLEEP" USING 1
           END-PERFORM
           .
```

You can use these fields to navigate through the Event Editor, to list the code paragraphs already associated with a given screen element, or to create a new paragraph associated with the screen element.

14.8.1 Entering the Event Editor

When you are working in the Screen Designer, the most straightforward way to use the Event Editor is often through the Event tab of the Property window. The Event tab lists all of the event, exception, and embedded procedures associated with the selected screen element type, as well as any AcuBench code insertions points defined for the screen element. These procedures and insertion points are discussed in the next section.

To enter the Event Editor from the Event tab of the Property window:

1. Select a screen element in the Screen Designer.
2. Expand the Event Procedure or Exception Procedure tree, if necessary, and select a procedure type.
3. Click in the Value column next to the procedure type, then click the browse ("...") button.

An Add Paragraph dialog opens in the foreground, while the Event Editor opens in the background.

4. Enter a name for the paragraph you plan to create, then click **OK**. To enter the Event Editor without creating a paragraph, click **Cancel**.
5. The fields at the top of the screen show the selected control, procedure type, and paragraph name. You can now begin entering code.

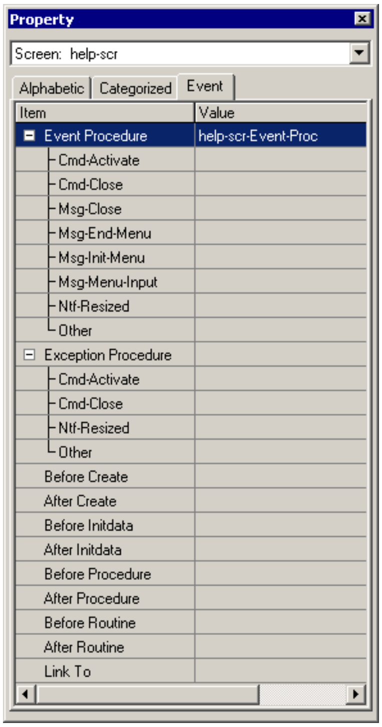
You can also enter the Event Editor at any time by double-clicking the Event Paragraph node for any program in the Structure view. If you open the Event Editor using this method, the fields at the top of the screen remain blank.

In some cases, you can enter the Event Editor by double-clicking a control on a screen form in the Screen Designer. In these cases, the Add Paragraph dialog appears, indicating the default paragraph type associated with the selected control. Edit the paragraph name as desired and click **OK** to enter the Event Editor. (If you click **Cancel**, no paragraph is created, but you still enter the Event Editor.)

14.8.2 Event Procedures, Embedded Procedures, and Code Insertion Points

The procedures and code insertion points listed on the Event tab of the Screen Designer Property window vary according to the type of control selected. Display-only controls, such as bars, frames, and entry fields, for example, do not have any procedures or code insertion points associated with them. In general, the more complex the control (and therefore the more ways a user can respond to a control), the more event and exception procedures can be associated with a control.

If you select a screen, for example, and expand the list in the Event tab of the Property window, you see the following procedures and insertion points:



Item	Value
⊟ Event Procedure	help-scr-Event-Proc
Cmd-Activate	
Cmd-Close	
Msg-Close	
Msg-End-Menu	
Msg-Init-Menu	
Msg-Menu-Input	
Ntf-Resized	
Other	
⊟ Exception Procedure	
Cmd-Activate	
Cmd-Close	
Ntf-Resized	
Other	
Before Create	
After Create	
Before Initdata	
After Initdata	
Before Procedure	
After Procedure	
Before Routine	
After Routine	
Link To	

The code insertion points are unique to AcuBench screens. These procedures (Before Create, After Create, Before Initdata, After Initdata, Before Routine, and After Routine) can be defined and inserted around generated code paragraphs. Before Create code that you define, for example, is performed just before the AcuBench-generated DISPLAY code for the screen. You might insert a paragraph here to load the data you want to see when the screen is displayed. After Initdata code is performed just after the AcuBench-generated "Initdata" code used to define the data structure for complex screen controls like combo boxes, grids, and tree view controls. You might insert a paragraph here to manually load data into any of these controls.

Event, exception, and embedded procedures, on the other hand, can be assigned to ACUCOBOL-GT screens and controls regardless of whether or not you are working in AcuBench. These procedures follow the structure and rules discussed in Chapter 6 of Book 2 and Chapter 5 of Book 3 in the ACUCOBOL-GT manual set.

One special AcuBench embedded procedure type is the Link-To paragraph. A Link-To paragraph is a way to assign a specific exception value to certain screen elements. When a user clicks a screen element that has a Link-To value assigned, the specified exception is sent to the program, which then performs the procedure associated with that exception.

Both Link-To and Cmd-Clicked procedures are invoked when the user clicks on a screen element. In the former case, a manually-assigned exception value is generated, updating the Key-Status data structure and invoking a procedure associated with the specified value.

```
WHEN Key-Status = 211
   PERFORM Client-Scr-Pets-Pb-Link
```

When a Cmd-Clicked exception procedure has been assigned to a screen element and a user clicks the control, an Event-Occurred message (exception value 96) is sent to the program, which then determines which control generated the exception, which event actually occurred, and whether or not a procedure exists to respond to that event.

14.9 Working with Screen Templates

When you create a new screen in the Screen Designer, the default New Screen interface provides you with a list of default screen templates (like Standard Graphical and Blank Character). In addition to these defaults, you can create and use your own screen templates. When you create screen templates, you have the option to either add them to the New Screen dialog (as described in **Chapter 13, section 13.4, “Adding Screen Templates”**), or to access them with the Add Screen command.

The process used to create a screen template file (“.stf”) is very straightforward. First, use the Screen Designer to create a screen, as usual. When you are finished, right-click on the screen form and select **Generate STF Document**. In the Save As dialog box, navigate to the folder where you want your template stored.

If you elect not to add your template to the New Screen dialog, you can use the Add Screen command to create a new screen based on a template. To do this:

1. In the Structure view, right-click the Screen icon and select **Add Screen**.
2. In the Add Screen to Program dialog, navigate to the directory containing your screen template, select the “.stf”, and click **Open**.

   The new screen opens in the Screen Designer.

When you use the New Screen interface to select a template and create a new screen, you have the option to change the screen’s name and unique prefix at the start. When you use the Add Screen command, however, your new screen is assigned the same name and unique prefix that you used when you designed the template. As a result, your first step after using the Add Screen command should be to change the screen name in the Property window. This will also update the screen’s unique prefix to match up to the first ten letters of the screen name. To assign an alternative unique prefix, right-click the screen form and select **Change Prefix**. When you change a screen’s unique prefix in the Change Prefix dialog, you can choose whether or not the changed prefix is assigned to existing controls.

14.10 Generating a Screen

As discussed in **Chapter 4, section 4.6.2, “Program Tag Options,”** when you create an AcuBench program that includes a program structure file, AcuBench adds tags to the source (“.cbl”) file to indicate the placement of generated code. When you design a screen, then use the Build/Generate command to generate COBOL code, AcuBench uses these tags to add the Screen Section, Working-Storage, and Procedure Division code needed to display the screen and make it function. If you are using the default code generation options (generating code into multiple COPY files), you will see several COPY statements added to the source file. These refer both to AcuBench-generated COPY files and to the ACUCOBOL-GT definition files used for screen handling.

Depending on the elements that you have added to your screen, the generated COPY files may include a Screen Section COPY file (".scr"), an event paragraph file (".evt"), and a menu paragraph file (".mnu"). The code generation process will also add screen DISPLAY and ACCEPT information to the Procedure Division file (".prd"), and update the Working-Storage file (".wrk").

AcuBench generates the Procedure Division code needed to DISPLAY and ACCEPT each screen in your program. This code is invoked through an "Acu-*screen*-Routine" paragraph. AcuBench automatically generates the code needed to invoke this routine for the program's main screen at startup. You add the code needed to perform the appropriate screen routine for each of the other screens in your program.

See **Section 14.11, "Testing Screens,"** for information about invoking secondary screens in a program without the need to write code.

14.11 Testing Screens

When you want to view and interact with the screen you have designed, use the **Build/Execute** command. If there are multiple screens in your program, AcuBench generates the code to execute just one of them. Although Screen Section entries and DISPLAY statements are added to the generated code for every screen, only the first screen's code is executed automatically. To test additional screens, you have two options:

- Add code, associated with a push button or other control, to call the remaining screens in your program:

  ```
  PERFORM Acu-LookupScreen-Routine.
  ```

- Use the Program Properties interface to change the main screen for the program. AcuBench generates the code to execute whichever screen is currently set as the main screen.

To change the default screen displayed at runtime:

1. Right-click the program node in the Structural view and select **Properties**.
2. Under “On program startup, set main screen to,” select a screen from the drop-down list.
3. Click **OK** to save your changes.

   When you execute your program, the selected screen is displayed.

14.12 Creating Portable Screens with AcuXUI

AcuBench includes support for AcuXUI technology, which allows you to display graphical screens in non-Windows environments. If you have purchased AcuXUI, you can design graphical screens in the AcuBench Screen Designer and enable AcuXUI at runtime.

To test your screens in an AcuXUI environment:

1. Expand the Build menu and select **Use AcuXUI**.

   AcuBench’s AcuXUI support can be used both when you execute local programs and when you execute programs remotely using the thin client.
2. Expand the Project menu and select **Settings** to open the Project Settings interface, then select the Environment tab.

   Two environment variables—CLASSPATHDIR and XUIJAR—are automatically added to the Environment tab when you open a new project. If desired, you can add an XUIPARAMS variable to define any Java parameters that you want to pass with the Java command line.
3. Make changes to the two AcuXUI-related environment variables as needed. Refer to the *AcuXUI User’s Guide* for more information.
4. Click **OK** to save your changes, then execute your program. The command line used to execute with AcuXUI appears in the AcuBench Output window.

If you are executing the program locally, the command line looks something like this:

```
javaw com.acucorp.acuxui.AcuXUI
   --acucobolgt "runtimepath" <options> "programpath"
```

If you are executing with the thin client, the command line looks something like this:

```
javaw com.acucorp.acuxui.AcuXUI -s <server>
   -p <port> -r <runtime options> alias
```

Note that there are differences between a graphical user interface displayed in the Microsoft Windows environment and one displayed with AcuXUI in the Java runtime environment. Some screen and control properties that the Windows environment supports, for example, are not supported by the AcuXUI environment. Screen and control size and spacing measurements also differ between the two environments.

For more information about the differences between the two graphical environments, as well as recommendations for optimizing screens created for use with AcuXUI, please refer to the *AcuXUI User's Guide*.

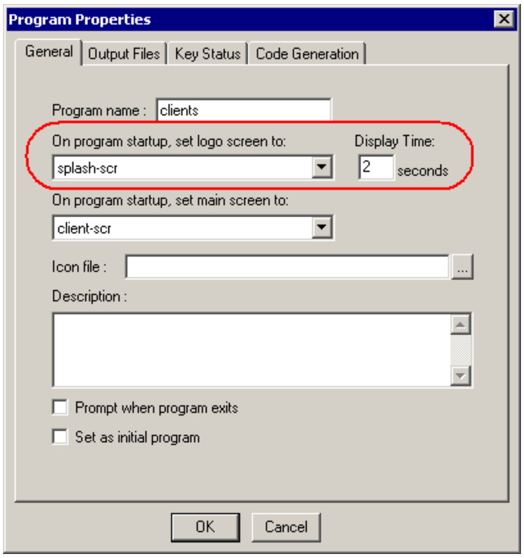
14.13 Creating a Logo Screen

You have the option to define a logo screen—often called a splash screen—for each AcuBench program that you create. A logo screen is displayed briefly before the program's main screen is displayed. As its name implies, a logo screen is often used to display a company or application logo.

When you design a logo screen in the Screen Designer, AcuBench can generate the code to automatically display the screen for a specified period of time before showing the main screen. To select a logo screen:

1. Right-click the program node in the Structure view and select **Properties**.

2. On the General tab of the Program Properties window, under “On program startup, set logo screen to,” select a screen from the combo-box.



Program name : clients

On program startup, set logo screen to: splash-scr

Display Time: 2 seconds

On program startup, set main screen to: client-scr

Icon file :

Description :

Prompt when program exits

Set as initial program

3. Use the “Display Time” entry field to specify how long the logo screen will be displayed before the main application window appears. You can specify a value between 1 and 300, indicating a display time between 1 and 300 seconds.

4. Click **OK**.

14.14 Creating Dual User Interfaces

It is possible to have both character-based and graphical screens in the same program. Moreover, for each screen in your user interface, you can have both a graphical and a character-based version, each with the same name. When two versions of a screen with the same name appear in the same application, the runtime chooses which to display based on the capabilities of the runtime environment.

To facilitate the process of creating a dual graphical/character interface for your programs, the Screen Designer allows you to copy screen items between graphical and character-based screen forms. You can lay out a screen on one type of screen form, then copy and paste all of the controls onto the other type of screen form.

When you copy and paste controls between screen types, however, there are some considerations to keep in mind:

- Graphical screen elements use pixels for positioning, while character screen elements use cells. This means that when an item is moved from one screen type to the other, the positioning will not remain exactly the same.
- Graphical screens can contain more types of screen elements (including toolbars, bitmaps, and so on) than character screens. Graphical screen elements cannot be copied onto character screens.

15 Controls, Menus, and Toolbars

Key Topics

Introduction 15-2
Standard Controls 15-3
ActiveX Controls 15-20
Using the Menu Designer 15-23
Using the Toolbar Designer 15-29

15.1 Introduction

This chapter provides more detail about the controls and other screen elements that can be used to create a user interface in the AcuBench Screen Designer. It starts with an overview of the ACUCOBOL-GT standard controls, then discusses the basics of ActiveX controls. Next, it talks about the process of adding menus and toolbars to your screens. (Both character-based and graphical screens may include a menu; only graphical screens can include a toolbar.)

The information contained in this chapter is supplemental to the information in Books 2 and 3 of the ACUCOBOL-GT documentation set pertaining to menus, graphical controls, and toolbars. In most cases, this chapter focuses on the AcuBench-specific aspects of using the various screen elements, although some more general principles are included.

15.1.1 Properties of Screen Elements

All components of a screen element are *properties*. When we talk about screen controls, these properties are classified into two groups: common properties and special properties. Common properties apply to many types of controls, and include:

size	title	value
color/intensity	font	visible/invisible
enabled/disabled	ID	

Note that some common properties have a special meaning (or no meaning) for some control types. (A bar control, for example, wouldn't have a title, value, or font.)

Each control also has its own set of special properties. Special properties give a control a special attribute or capability. For example, an entry field can have the special properties Max-Text or Max-Lines, which determine how many characters or lines of information users will be permitted to enter in that field. All special properties require a value. Max-Text, for example, takes a numeric value that specifies the maximum number of characters that can be entered in the field. Some special properties are used by more than one control type. For example, the special properties that apply to bitmap buttons are also used by the push button, radio button, and check box controls.

Styles are a special type of property that affects the appearance or behavior of a control. For example, some of the styles that apply to a radio button include Bitmap, Framed, and Notify. Styles do not take a value. Most styles apply to only a certain type of control, although a few are common to all controls.

The process of setting control properties is discussed in **Chapter 14, section 14.5, "Configuring Control Properties."**

15.1.2 Events Overview

In addition to setting appearance and behavior properties of a screen item, you can associate code with the item to determine its functionality. Just as each screen element is associated with a certain set of properties, it is also associated with a certain set of events. An *event* is a user action to which the screen item is capable of responding. If a user enters information into an entry field, for example, the entry field is capable of recognizing each keystroke that the user types as a Notify-Changed event. Likewise, a push button control can recognize a mouse click as a Command-Clicked event. You can choose to capture these events and write code to respond appropriately to the user action.

In AcuBench, the Event tab of the Property window provides an interface that both lists the events to which a given control can respond and allows you to enter code to handle that event. For more information about adding event code in AcuBench, see **Chapter 14, section 14.8, "Associating Code with Screen Elements."** For a more general discussion of events and event handling, see Book 2 of the ACUCOBOL-GT documentation set.

15.2 Standard Controls

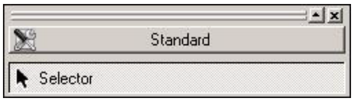
When you are working in the Screen Designer, the Screen Component Toolbox lists the standard and ActiveX controls available for use. If you are working with a graphical screen, all of the standard controls are available for you to use. If you are working with a character-based screen, those controls that are available only with graphical windows appear disabled in the Screen Component Toolbox. The sections that follow provide summary descriptions of each of the standard controls.

When you are working with ActiveX controls, the list of controls that appears in the ActiveX portion of the Screen Component Toolbox depends on which controls you have installed and registered. A detailed description of an ActiveX control can usually be found in the vendor documentation that accompanies it. More information about programming with ActiveX controls can be found in Chapter 3 of *A Guide to Interoperating with ACUCOBOL-GT*.

The size and location of most controls on a screen form can be changed. To change the size, click the control to select it, then drag a corner of the "handle" that appears. You can also use the keyboard shortcuts **Shift+Arrow** and **Ctrl+Shift+Arrow** to change the size of a control or window. To change a control's location, click on the control and drag it to the desired position on the screen form. You can also use the arrow keys or the **Ctrl+Arrow** keyboard shortcut to move items on the screen form. Note that status bar size and location cannot be changed. A status bar automatically grows and shrinks horizontally to match the width of the owning window.

15.2.1 Selector

When you first open the Screen Designer, the Selector is highlighted in the Screen Component Toolbox and the mouse pointer is a white arrow.

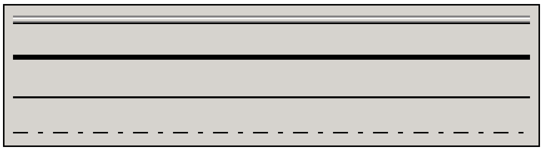


When you select any control in the Toolbox and move the mouse to the screen form, the mouse pointer becomes a crosshairs. After you draw a control on the screen, the mouse pointer returns to being an arrow, and the Selector icon appears highlighted in the Toolbox. When the Selector is chosen, you can select, move, and resize controls already drawn on the screen form.

If you have selected a control type in the Screen Component Toolbox and decide that you don't want to draw that control on your screen, you can cancel your selection by clicking the Selector icon.

15.2.2 Bar

The bar control allows you to draw a horizontal, vertical, or diagonal bar in the screen form at design time.



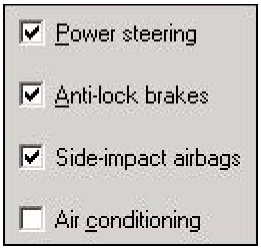
15.2.3 Bitmap

Available only in graphical environments, a bitmap is a graphic ("*.bmp", "*.jpg", "*.jpe", or "*.jpeg") that can be added to a screen in order to enhance its appearance, for example, or to add a company logo. A bitmap can also be added to a push button, radio button, or check box control.

A bitmap control is rectangular and non-transparent by default. To display a bitmap so that it appears non-rectangular, or to make sure that some portion of a bitmap matches the background color of the screen, use the Transparent Color property. In the Property window, click the button next to the Transparent Color property to open the Choose Transparent Color dialog box. As you move your mouse over the bitmap, the dialog box shows the RGB value of the current color. When you click a spot on your bitmap, the color of that spot is set as the transparent color, and when the program executes, that color does not appear (in other words, the background color of the screen shows through). For more information about bitmaps and the Transparent Color property, see Chapter 5 in *ACUCOBOL-GT User Interface Programming*.

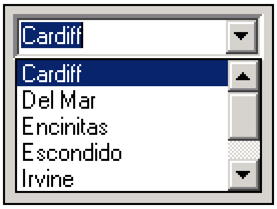
15.2.4 Check Box

A check box control includes a selection box and a text label. When a user selects the control, a check mark appears in a check box. Typically, check boxes are used when you need to provide the end user with a limited number of choices from which any number may be selected, or when you want to present the end user with an option that may be turned on or off. Check boxes are also used to give the end user a Yes/No or True/False option.



15.2.5 Combo Box

A combo box is an input control that combines a list box and an entry field. Typically, it contains several options in the list portion of the combo box, and the end user can select the needed option with the mouse pointer, or write the needed option in the entry field portion of the combo box. You can create three kinds of combo boxes: drop-down (users can type in the entry-field portion of the control and use a button to reveal the list, which is otherwise hidden), drop-list (users must make a selection from the drop-down list and cannot type in the entry-field), and static list (the list is permanently displayed on the screen).



When you draw a drop-down or drop-list combo box control in the Screen Designer, remember to include space for the data in the list. In other words, do not draw the control exactly the size of the entry-field portion of the control. Instead, select the control and drag the selection handle downward, leaving an “empty” space under the control. If this extra space overlaps other controls on the screen form, when users expand the list at runtime, it temporarily overlaps the other control.

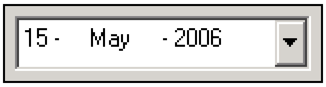
15.2.6 Date Entry

Available only in Microsoft Windows environments, the date entry control provides a convenient way for users to enter date or time values. Depending on the format that you choose, the date entry may include a graphical calendar from which users can select a date.

Four standard date and time formats are listed for the Display Format property in the property sheet for the control. You can also enter a custom Display Format for the control, combining various date and time elements.

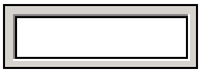
Note: Although the property sheet lists a Short-Date value format of YYMMDD, the actual Short-Date format is dependent on Windows operating system settings. See Chapter 5 of the *ACUCOBOL-GT User Interface Programming* guide for more information.

Some restrictions apply to the use of this control. See Chapter 5 of *ACUCOBOL-GT User Interface Programming* for details about the styles and properties available for this control.



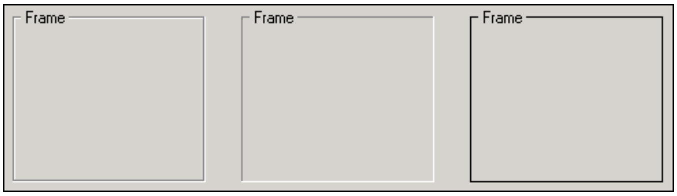
15.2.7 Entry Field

An entry field control is used for accepting data from the end user, such as multiple lines of text. Typically, it is the field where an end user enters information, such as a name and address. You can create either single line or multiline entry fields in the Screen Designer.



15.2.8 Frame

A frame control is a “box” used for grouping related controls together. You can put controls into frames to visually separate them from other controls or to group related controls together. For example, it is a common practice to enclose radio button groups within a frame.



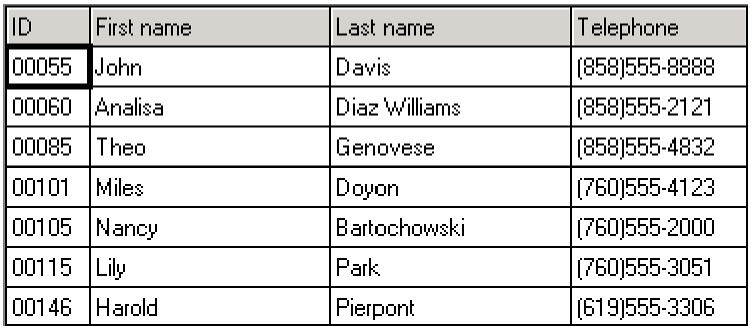
When you draw controls directly on a frame, you cannot drag them out again. They are grouped inside of it. This allows you to move a frame and have the control(s) remain inside of it.

When you create a frame first and then add controls into it, you can copy and paste those controls within it. Note that if you select this frame and delete it, the controls are also deleted. If you create controls first, and then put a frame around them, you cannot select the controls within that frame to copy and paste or move them, unless you move the frame away from the controls. You can delete this frame without deleting the controls inside.

When users tab through controls on a screen, the order of the cursor movement depends on the tab order established for those controls. When you create controls inside a frame control, the tab order of the frame control itself is part of the tab order for the entire screen, but items within the frame have their own, distinct tab order. The tab order for controls inside of each frame starts at "1" in the Property window. To change the tab order of controls within a frame, right-click the frame and select **Tab Order**.

15.2.9 Grid

Available only in graphical environments, the grid control is a two-dimensional table of data fields. Each element of this table, called a "cell," can hold text or a bitmap, or both. Grids are relatively complex controls with many properties that you can use to customize their appearance and behavior.



ID	First name	Last name	Telephone
00055	John	Davis	(858)555-8888
00060	Analisa	Diaz Williams	(858)555-2121
00085	Theo	Genovese	(858)555-4832
00101	Miles	Doyon	(760)555-4123
00105	Nancy	Bartochowski	(760)555-2000
00115	Lily	Park	(760)555-3051
00146	Harold	Pierpont	(619)555-3306

Grids are organized into rows, columns, and records. In a grid, a "row" is a grouping of cells that appear on one line in the control. A "record" is one or more rows that are treated as a logical unit. A "column" identifies a particular cell in a record. By default, a record occupies one row in a grid, but you can arrange for a record to "wrap around" to the next row when it passes the right edge of the grid. When this occurs, a record occupies more than one row in the grid. You might want to construct a grid like this when you want to see many fields in a data record at once. Alternatively, you can have the grid use scroll bars to access cells past the right edge of the control. Column, row, and record numbers all start at "1."

Grids come in two formats: with horizontal scrolling and without. When you opt for horizontal scrolling, each record may occupy only one row in the grid. Grids with horizontal scrolling appear much like a spreadsheet. Without horizontal scrolling, a record may occupy more than one row. In either case, vertical scroll bars appear automatically when needed.

The grid operates in two different modes: navigate mode and entry mode. While the grid is in navigate mode, the arrow keys move the cursor around the grid. This mode is the default. The grid shifts to entry mode when the user starts to modify data. In this mode, the arrow keys are used to edit the current cell's data. When the user finishes a cell, the grid returns to navigate mode.

The exact set of keys understood by the grid depends on the host system. Under Windows, the following keys are used in navigate mode:

Up/Down arrow	Moves the cursor to the same column in the previous/next record	Tab (with USE-TAB)	Moves the cursor right, wrapping to the next record when at the last cell in the record
Left/Right arrow	Moves the cursor to the previous/next column in the record	Backtab (with USE-TAB)	Moves the cursor left, wrapping to the previous record when at the first cell in the record
Home	Moves to the first column in the record	End	Moves to the last column in the record
Ctrl+Home	Moves to the first column of the first record	Ctrl+End	Moves to the first column in the last record
Page up/down	Moves the cursor up/down one page	Enter	Shifts to entry mode for the current cell; highlights the contents for editing
Shift+Enter	Moves the cursor to the first column of the next record	Any printable character	Shifts to entry mode for the current cell; overwrites the contents with the character

When the grid is in entry mode, the user types into an entry field control. All of the editing keys usable by an entry field are usable here. The user leaves entry mode by typing **Enter** or **Tab**/**Backtab**, or by clicking on another cell with the mouse.

Clicking a mouse on a cell moves the cursor to that cell. Clicking on the selected cell shifts to entry mode. So double-clicking on any cell has the effect of modifying that cell's contents. The first click moves the cursor and the second click shifts modes.

You should note that the Property window Value cell for the Row-Dividers property accepts only a single digit. You cannot enter multiple values in the workbench for Row-Dividers.

Paged grid

Sometimes you may want to use a grid control to view a large number of records. When this is the case, a normal grid is impractical. Just loading all the records into the grid could take an excessive amount of time. To handle this case, the grid supports a Paged style. When you use this style, the grid holds only as many records as can be seen on the screen. This is called a "page" of data. Four buttons that get the next/previous record and the next/previous page of records replace the vertical scroll bar. When the user clicks one of these buttons, the control sends a message to the program asking for the appropriate data. The program itself must supply the needed data. This data typically comes from an indexed file, and the appropriate program logic is to do one or more READ NEXT/PREVIOUS statements to retrieve the data.

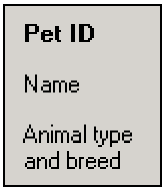
The grid control communicates requests for more data via events. More information about grid events can be found in Chapter 6 of *ACUCOBOL-GT User Interface Programming*.

Paged grids never hold more data than they can display. When adding a record to a full page, the control deletes the top non-heading record when a record is added to the end of the grid. This causes the grid's contents to scroll upwards. When adding a record to any other position, the last record in the grid is deleted. This causes all records after the one being added to scroll downwards.

Note that the current cell is not changed when the grid is paged. In other words, if the grid's cursor is at row 2, column 3, it is at row 2, column 3 after the user clicks the "next record" button. This action effectively moves the cursor to a new record, even though its physical location has not changed. Unlike other forms of cursor movement, this does not generate any additional events. If you are performing special actions when the cursor enters a new cell (for example, displaying related information outside of the grid), then you should perform these actions in response to paging events as well as cursor-movement events.

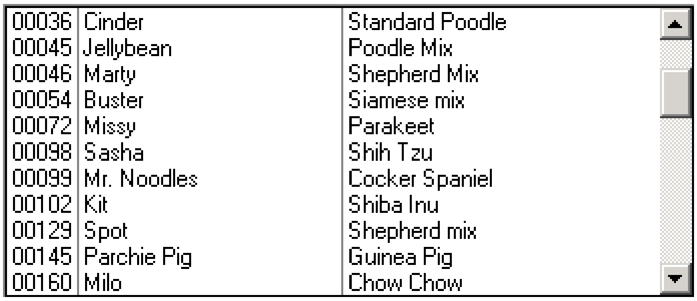
15.2.10 Label

The label control allows you to display a simple string of text on the screen form. This text cannot be changed by the end user, though it can be changed by the application at runtime. The label control is typically used to display permanent labels for entry fields, explanatory notes, or titles. It can also be used to show changing activities, such as the time or the progress of a file-copying activity. Labels may occupy multiple lines. When a label is displayed on multiple lines, it uses word-wrapping, if possible, so that words are not broken across lines.



15.2.11 List Box

A list box is a rectangular list from which an end user can select one or more items. A list box does not allow direct data input. If the list box contains many items, the user can navigate through it with a scroll bar.



00036	Cinder	Standard Poodle
00045	Jellybean	Poodle Mix
00046	Marty	Shepherd Mix
00054	Buster	Siamese mix
00072	Missy	Parakeet
00098	Sasha	Shih Tzu
00099	Mr. Noodles	Cocker Spaniel
00102	Kit	Shiba Inu
00129	Spot	Shepherd mix
00145	Parchie Pig	Guinea Pig
00160	Milo	Chow Chow

Paged List Box

The standard list box control provides a convenient way for a program to implement a look-up facility for a group of items. It is also tempting to extend this type of use into a method for locating records in a data file. Unfortunately, this doesn't work well when the file contains too many records. The programmer runs into two main problems:

1. The standard list box has a limited capacity, usually fewer than 2000 items.
2. It takes too long to load the list box with the entire set of items.

Also, if the number of items is very large, the user may have a difficult time locating a particular item. There are two reasons for this:

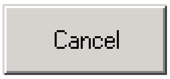
1. The resolution of the scroll bar's slider is too coarse.
2. The search mechanism is too primitive (for example, a single-character match on the first byte of the record).

The paged list box is a variation of the standard list box that solves all of these problems. A paged list box works by managing only a limited number of records at a time. When it needs more records, it requests them from the controlling program. Paged list boxes are intended to be used in conjunction with a large, ordered data source, typically records stored in an indexed file.

More information about how to use paged list boxes can be found in Book 2 of the ACUCOBOL-GT documentation set.

15.2.12 Push Button

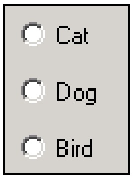
A push button is used to carry out an action when clicked by the user. It typically returns a unique value when selected, and its border changes, indicating that it has been selected. A common use for a push button is an "OK" or "Exit" button.



15.2.13 Radio Button

A radio button is used to select an option, usually from a group of radio buttons. Unlike a check box, only one radio button in a group may be selected at a time. When a radio button is selected, it displays a black center to indicate that it has been selected. Typically, a group of radio buttons would be used to indicate a choice, such as a payment method (for example, cash, check, or credit card).

Because radio buttons always function in sets, if you want multiple sets of radio buttons in a screen, you should draw each set of buttons inside a frame control. When a set of radio buttons is inside a frame, clicking one of the buttons clears the other buttons in the set.



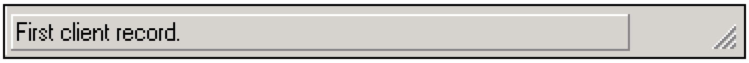
15.2.14 Scroll Bar

The scroll bar allows end users to change the viewing area in a list or on a form. Scroll bars can also be used to move through a range of values by increments. Typically, a scroll bar is either a vertical or horizontal slider bar with arrows attached at each end.

15.2.15 Status Bar

Available only in graphical environments, the status bar control allows you to create a status bar at the bottom of your screen. You can define up to 128 panels in your status bar, depending on how many items you want to track on your screen. Each screen may have only one status bar.

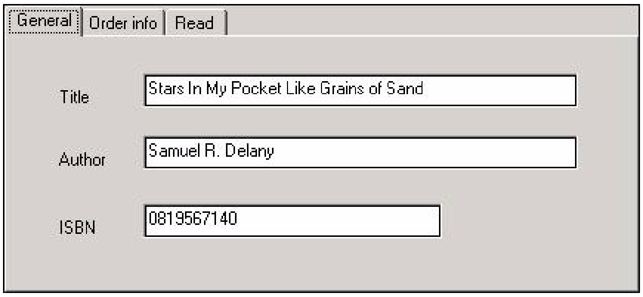
You define the panels in a status bar in the “Panel description” area of the Panel Setting dialog box. To display this dialog, click in the Panels’ Setting value column in the status bar control Property window.


First client record.

15.2.16 Tab

Available only in graphical environments, the tab control combines a box with a tab for a control that looks like a file folder. The user clicks on any tab to bring it forward. The program typically places different screen elements in the box, depending on the tab selected.

Tabs can be oriented horizontally at the top or bottom of the control, or they can be oriented vertically at the left or right of the control, via the control’s Tab Orientation property. Note that if the tab page is filled to its outer edges, you may have to make slight resizing or repositioning adjustments when you change the tab orientation.



Title: Stars In My Pocket Like Grains of Sand
Author: Samuel R. Delany
ISBN: 0819567140

When a user clicks on a tab, the program is informed of the new selection and the tab's appearance is updated. The behavioral distinction between tabs and push buttons is that a tab responds immediately when clicked, and a push button responds with the "clicked" event only when the mouse button is released. You may allow the user to activate the tabs with the keyboard by accepting the tab control as any other control, but you need not do so if you wish to provide only a mouse interface.

When you create a tab control in the Screen Designer, there are a few special features to keep in mind. First, note that you can select either the entire tab control (the highlight frame and handles outline the control) or a page of the control (the highlight frame and handles define a rectangular region that does not include the "tab" portion of the control).

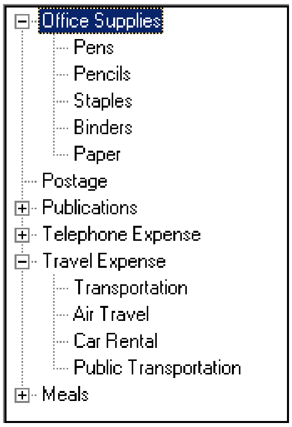
Note also that when you have selected a tab control or one of its pages, the Property window indicates the nature of the current selection. If the entire control is selected, the Property window entry begins with "Tab Control" (followed by the control name). If a tab control page is selected, the Property window entry begins with "Page" (followed by the page name).

The tab control uses the following special editing commands:

- To add a new page to the tab control, make sure that the entire control is selected, then right-click and select **Insert New Page** (near the bottom of the right-click menu).
- Once your control contains multiple pages, you can select **Next Page** or **Previous Page** from the right-click pop-up menu to navigate the control. If the entire control is selected, you can also switch between pages of the control by clicking the tab portion of the control.

15.2.17 Tree View

The tree view control presents hierarchical data in a list. This list is indented to show the "parent" and "child" relationships in the data. Users can "expand" or "collapse" items in the list to view or hide subsidiary items.



In a tree view control, an ID is assigned to each element when it is added to the control. This provides a unique way to identify each item and allows for duplicate items at different points in the hierarchy without any confusion. Tree view IDs are declared in Working-Storage as USAGE POINTER data items. Tree View Designer functions are explained in detail in the following sections.

Using the Tree View Designer

Create a tree view for a screen by dragging a tree view control from the Screen Component Toolbox to any location in the screen form. When you add a tree view control to the screen, the Tree View Designer dialog box appears. This dialog box lets you add, delete, and move tree view items.

Creating a tree view for a screen

1. In the Screen Component Toolbox, click the tree view control icon.

2. Move the pointer to the screen form. The pointer changes to a crosshair.

3. In the screen form, drag the left mouse button to add a tree view control.
4. If the Tree View Designer does not appear automatically, double-click the tree view control box you have placed on the screen form. The text marker is highlighted in the dialog box.

The tree view control has two Property windows. The first one you see is available when you enter tree information in the Tree View Designer. It governs the individual "tree" and "children" properties. The second Property window is more extensive, and is available when you enter the tree and children information. It governs the more global behavior of the tree view control.

Adding a tree item to a tree view

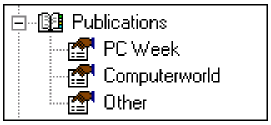
1. In the Tree View Designer, double-click the text marker. A new text field appears in place of the text marker at the left side of the dialog box. If the tree view already has tree items, the new item appears below and at the same level as the selected tree item.
2. If you want to add a tree item before a selected entry, right-click in the designer and select the **Insert Item Before** command.
3. Enter a name for the new tree item from the keyboard. Press **Enter**.

Adding a child item to a tree view

1. Highlight the tree item that needs a submenu and click the **Insert a sub-item** button. You can also right-click in the designer and select the **Insert Sub-item** command. The cursor drops to a lower field, enabling you to type in the name of a child item.
2. Once you have entered all of the child items for a particular tree item, you can enter the next tree item and child items by selecting the text marker below the last child item.

Adding bitmaps to a tree view

The Tree View Designer makes it easy to add bitmap icons to items in a tree view control.



1. In the Screen Designer, select the tree view control on the screen.
2. In the Property window, click the Value column next to the Bitmap property, then click the browse button.
3. Navigate to the location of the bitmap or bitmap strip that you want to use, then click **Open**.
4. In the Tree View Designer, select an item in the tree, then use the Property window to enter a value for the Bitmap Number property. If you are using a single bitmap image, rather than a bitmap strip, this number is “1”. If you are using a bitmap strip, enter the number corresponding to the position of the icon that you wish to use.

Deleting an item from a tree view

1. In the Tree View Designer, select and highlight the tree item or child that you want to delete.
2. Click **Delete**.

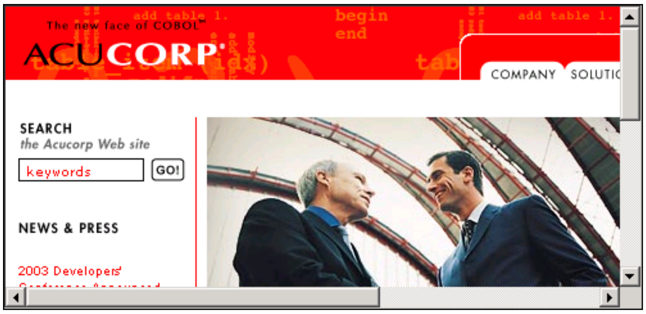
Moving an item in a tree view

1. In the Tree View Designer, select the tree or child that you want to move.
2. Click the **Move item up** or **Move item down** button, depending on where you want the item to move. You can also right-click in the designer and select the **Move Up** or **Move Down** command.

15.2.18 Web Browser

Available only in graphical environments, the Web browser control uses the runtime to browse the Web; invoke e-mail, telnet, and FTP services; and display multimedia and other file types. It also allows users to view Windows folders and files. It is used just like any other ACUCOBOL-GT control.

The Web browser control is installed with Microsoft Internet Explorer. It allows the user to view the main window of the Microsoft Internet Explorer on a screen created by the Screen Designer. It provides the functionality for displaying Web pages containing HTML, scripting, and ActiveX controls and Java applet content.



15.3 ActiveX Controls

AcuBench makes it easy to include ActiveX controls in graphical programs designed to run in Microsoft Windows. Once you have installed the control on your system, you can add it to the Screen Component Toolbox, then place it on your screen form just like a standard control.

Note that ActiveX controls that you include in your program must be installed and licensed in accordance with the vendor's requirements both on your development system and on every system on which the program is deployed. It is up to you to know how to program and distribute the ActiveX controls that you use. Refer to the documentation provided by your ActiveX control supplier.

General information about how to use ActiveX controls in your application can be found in Chapter 4, “Using ActiveX Controls and COM Objects in Your COBOL Program,” in *A Guide to Interoperating with ACUCOBOL-GT.*

For your convenience, several Microsoft controls have been included with the ACUCOBOL-GT Windows runtime. You can use these controls in your program and redistribute them to your end users as needed. These controls are installed and licensed automatically, and are also found on your product CD in the “ms” directory. For more information about these controls, see section 3.4.9, “Distributing Applications Containing ActiveX Controls,” in *A Guide to Interoperating with ACUCOBOL-GT.*

15.3.1 Adding ActiveX Controls to the Component Toolbox

You can add an ActiveX control to the Screen Component Toolbox as follows:

1. Make sure that the control that you want to use is installed and registered on your system. If the control is installed but not registered, you can use the Microsoft Windows **regsvr32** command to add the control to the Windows registry. The syntax is:

   ```
   regsvr32 control-name.ocx
   ```

2. In the AcuBench Screen Designer, click the **ActiveX** bar in the Screen Component Toolbox. This hides the list of standard controls, opening an area that can be populated with ActiveX controls.

3. Right-click in the Screen Component Toolbox and select **ActiveX Control Components**. This opens the ActiveX Control Components window, which lists all of the ActiveX controls currently installed and registered on your system.

   This is the same list that appears when you use **AXDEFGEN**, the ACUCOBOL-GT ActiveX definition generator. AcuBench uses **AXDEFGEN** to perform necessary ActiveX-handling tasks behind the scenes. This means that you don’t have to access the utility directly when you are working in AcuBench.

4. Scroll through the list of ActiveX components to find the control(s) that you would like to use, then mark the check box next to the control(s).

5. When you are finished selecting controls, click **OK**.

   An icon is added to the ActiveX section of the Screen Component Toolbox for each control that you have selected.

Note: During code generation, AcuBench automatically adds an @ symbol to the ActiveX's control name. This avoids possible errors that can result from ActiveX control names that are identical to AcuBench control names, or to COBOL reserved words.

15.3.2 Using ActiveX Controls in Your Screen

The process of working with ActiveX controls in the Screen Designer is very similar to that of working with standard controls. To add an ActiveX control to a screen, do the following:

1. Select the control in the Component Toolbox, then draw it on the screen form.

   If you have an appropriate license, a graphical representation of the control appears on the screen form. If the control requires a design time license that you do not have, AcuBench shows a warning message, then draws a gray placeholder on the screen. In this case, the control will display normally at runtime, but you may have trouble changing control properties or coding the control. If this happens, see the control vendor for licensing information.

2. Once an ActiveX control is placed on a screen, its initial properties (state) can be configured in the Property window. In addition, most ActiveX controls include a special interface for property configuration. If the vendor has provided such an interface, you can access it by right-clicking the control on the screen form and selecting **ActiveX Control Properties**.

3. If you use an ActiveX control's special property interface to configure the control, AcuBench saves your settings in a resource file, "*program*.res". This file must be included when the program is deployed, or the control is displayed with its standard, default values.

15.3.3 Removing an ActiveX Control from the Component Toolbox

Use the following steps to remove a control from the ActiveX Screen Component Toolbox:

1. Click the **ActiveX** bar in the Screen Component Toolbox and then right-click in the toolbox. Select the **ActiveX Control Component** command from the pop-up menu. The ActiveX Control Components dialog box is displayed.
2. Find the item on the list that you want to remove and click in the adjacent check box to remove the check mark.
3. Repeat for other items that you want to remove.
4. Click **OK**.

All list items that do not have a check mark in the check box are removed from the ActiveX control group in the Screen Component Toolbox, but they are not removed from the Windows registry or uninstalled from the system.

15.4 Using the Menu Designer

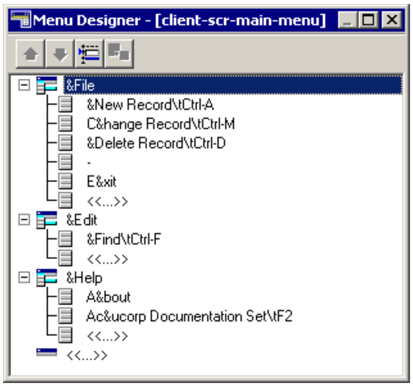
The Menu Designer is a tool that allows you to create a menu bar or pop-up menu for a screen. Menus typically consist of menu titles, menu items, and separator bars. When you double-click the Menu Designer icon on your screen, the Menu Designer window appears. The Menu Designer window is the same for both graphical and character-based screens. The Screen Designer lets you create multiple menus for a screen, but a screen can use only one menu bar (displayed at the top of the screen) at a time.

15.4.1 Creating a Menu

If you used the Standard Graphical or Standard Character template to create your screen, the Menu Designer icon appears on the screen by default. This default menu contains a single menu item (Exit) and is defined as the screen's main menu. (A main menu is displayed as a menu bar along the top of the window at run time.)

If you have created a blank screen, or are using another template that does not include a default menu, drag a Menu item from the Screen Component Toolbox to any location in the screen form. The Menu control icon does not appear when you run your screen, so its exact location in the screen form is not important.

When you draw a Menu icon on the screen form, the Menu Designer interface opens automatically. You can also enter this designer at any time by double-clicking an existing Menu icon on a screen form.


Menu Designer - [client-scr-main-menu]

```
&File
  &New Record\tCtrl-A
  C&hange Record\tCtrl-M
  &Delete Record\tCtrl-D
  -
  E&xit
  <<...>>
&Edit
  &Find\tCtrl-F
  <<...>>
&Help
  A&bout
  Ac&ucorp Documentation Set\tF2
  <<...>>
<<...>>
```

The Menu Designer interface lets you add, delete, and move menu items. It also provides an interface (through the Event tab of the Property window) for associating code with each menu item.

15.4.2 Building the Menu

To add, remove, or change items in your menu, use the following steps:

1. In the Menu Designer, double-click the menu text marker ("<<...>>").

   A text field appears in place of the menu text marker.

2. Enter the name of your menu item in the text field.

   To underline one character in the menu item name, indicating that the letter is a hot key, place the ampersand ("&") character before the letter you want underlined. (For example, "E&xit", indicates that the exit shortcut is Alt+X.)

3. Press **Enter** or **Tab** to complete the menu item.

   The text marker moves to the next line. You can now repeat the previous steps to create another menu item at the same level as the first, or use the steps that follow to add a menu subitem.

4. With a menu item selected, right-click and select **Insert a sub-item**. You can also use the **Insert a sub-item** button at the top of the Menu Designer window.

   The cursor drops to a submenu field, enabling you to type in the name of a submenu item.

5. Once you have entered all of the submenu items for a particular menu item, you can enter the next menu item and submenu items by selecting the menu text marker below the last submenu item.

   The Menu Designer allows a maximum of four levels of submenu items (five including the main menu).

6. If you want to add a menu item before an existing item, highlight the existing item and click the **Insert item before** button at the top of the Menu Designer window. You can also right-click in the dialog box and select the **Insert Item Before** command. The new item appears above and at the same level as the selected menu item.

7. To insert a separator between menu items, highlight a menu item, right-click, and select **Insert Separator** command. A separator is inserted before the selected menu item.

   Note that you can also create a separator by entering a single hyphen ("-") character as the text of a menu item. AcuBench automatically changes the Separator property of that item to TRUE, creating a separator.

8. If you want to remove an item from the menu that you are creating, highlight the menu item that you want to delete and press **Delete**.

9. To move a menu item up or down, select the item and drag it to the appropriate spot in the menu, or use the up and down arrow buttons at the top of the Menu Designer window.

15.4.3 Adding Menu Functionality

The process for associating code with menu items in AcuBench is very straightforward. As you add items to your menu in the Menu Designer, AcuBench automatically associates an exception value with each item. You can either use this default value or use the Property window to enter a different exception value. To associate a menu item with code, select the menu item, then use the Event tab of the property window to create a Link To paragraph for that item.

When you build a menu, it is a good idea to associate keyboard shortcuts with each menu command. There are two ways to do this:

1. Place the "&" character in front of a letter of the menu item in the Menu Designer. Users can invoke that menu item by pressing the Alt key and the designated letter.
2. Use a runtime configuration file to assign a keyboard shortcut to each exception value associated with a menu item.

When you use a runtime configuration file to assign a keyboard shortcut, the steps are as follows:

1. Add your item name in the menu designer, then enter the special character combination "\t" and the keyboard shortcut that you want to associate with the item. The result should look something like this:

   ```
   Add\tCtrl+A
   ```

   At runtime, the "\t" becomes a tab character, adding appropriate spacing between the menu item and its keyboard shortcut.
2. Make a note of the exception value associated with your menu item. If you prefer, you can change the Exception Value property associated with the item in the Property window.

3. To add a configuration file entry to tie the menu item to the keyboard shortcut that you have specified, open your configuration file. (In AcuBench, this generally means opening the Project Settings window, selecting the Runtime tab, and pressing Edit to open the Configuration File Editor.)

4. Add an entry for the KEYSTROKE configuration file entry that specifies the exception value and the keyboard shortcut that you want to associate. The entry should look something like this:

   ```
   KEYSTROKE Exception=1003 ^A
   ```

   In the example, when the user presses Ctrl+A, an exception of 1003 is generated. Any code associated with that exception (key-status) value is then executed.

   More information about the KEYSTROKE configuration variable can be found in Chapter 4 of Book 1 of the ACUCOBOL-GT documentation set.

5. Back in the menu designer, use the Event tab of the Property window to create a Link To paragraph associated with the menu item. This paragraph is invoked when a user selects the menu item or uses the keyboard shortcut associated with the menu item.

15.4.4 Enabling the Menu

The Screen Designer lets you create multiple menus for a screen. You determine whether the menu appears as a menu bar at the top of the screen or as a right-click pop-up menu. Note that while a screen can have only one menu bar (or main menu), you can assign pop-up menus to any control on the screen that is not a display-only control. So your might assign one menu bar and one pop-up menu for your screen, then have additional pop-up menus associated with a grid control and an entry field that appear on the screen. You can also assign the same pop-up menu to multiple screens or controls.

To add a menu bar (main menu) to your screen:

1. Select the Menu icon on the screen form.

2. In the Property window, make sure that the menu's Style property is set to **Static Menu**.

   Make a note of the Handle Variable property associated with the menu.

3. Select the screen form in the Screen Designer.
4. In the Property window, scroll to the **Main Menu** property.
5. Click in the Value field and expand the drop-down list, then select the handle of the menu that you want to use as your screen's main menu.

When you select a MAIN MENU handle, the generated code causes this menu to display at the top of your window. The code is put into the ".cbl" program generated. For example, if a screen is called form1 and the static menu handle is FORM1-MN-1-MENU, the code in "form1.cbl" is:

```
PERFORM BUILD-Form1-MN-1-MENU.
```

This code generates "form1.mnu", which contains:

```
CALL "w$menu" USING WMENU-SHOW, MENU-HANDLE.
```

Pop-up menus do not display via calls to W$MENU like this. They show up automatically when you right-click on the control. The code that is generated for the pop-up menu is just a single line of code in the "form1.prd" file, as follows:

```
    MOVE menu-handle TO Form1-MN-2-Handle
```

To define a pop-up menu for use with a screen or control, do the following:

1. Select the Menu icon on the screen form.
2. In the Property window, make sure that the menu's Style property is set to **Pop-Up Menu**.

   Make a note of the Handle Variable property associated with the menu.
3. Select the screen form or screen element in the Screen Designer.
4. In the Property window, scroll to the **Pop-Up Menu** property.
5. Click in the Value field and expand the drop-down list, then select the handle of the menu that you want to use as a pop-up menu.

15.5 Using the Toolbar Designer

The Toolbar Designer is a utility that allows you to design and create toolbar controls for a screen. The Toolbar Designer is available only for graphical screens. A toolbar is a collection of controls arranged in a row below a screen's menu bar (if present), or at the top of the window if no menu bar exists.

The size of a Toolbar Designer can be changed through use of the left mouse button. To change the size, click the box surrounding the Toolbar Designer and drag a corner of it until it becomes the size you need.

When you draw controls directly on a toolbar, you cannot drag them out again. They are grouped inside of it. This allows you to move a toolbar and have the control remain inside of it. You can copy a control that is in a toolbar and paste it outside of the toolbar if you want to remove it and use it elsewhere.

The Screen Designer lets you create multiple toolbars for a screen, and a screen can use more than one toolbar at a time. More information about using more than one toolbar at a time can be found later in this chapter.

15.5.1 Creating a Toolbar

You can use the following steps to create a toolbar for a screen:

1. In the Screen Component Toolbox, click the **Toolbar Designer** button.
2. Move the pointer to the screen form. The pointer changes to a crosshair.
3. Click anywhere in the screen form to add a Toolbar Designer window. By default, the window appears at the top of your screen form. Adjust the Toolbar Designer's properties in the Property window as desired. The Toolbar Designer generates no events.
4. Draw controls in the Toolbar Designer window. You can put any type of control in a toolbar except a menu or another toolbar. Controls should be placed in a single row, and they should not overlap.

15.5.2 Using More Than One Toolbar

The Screen Designer lets you create multiple toolbars for a screen and allows a screen to use more than one toolbar at a time.

To use multiple toolbars on a screen, create the toolbars, then place them in the order you want them to be seen on your screen, from top to bottom. When you run your screen, the toolbar you placed at the top appears at the top of the screen, below the menu bar. Below that toolbar appear any other toolbars you created in the top-to-bottom order that you placed them on the screen form.

16 Configuring the Report Composer

Key Topics

Introduction 16-2
Customizing the Report Composer Interface 16-2
Establishing Report and Control Defaults 16-4
Adding Report Templates 16-5
Report Keyboard Shortcuts 16-7

16.1 Introduction

The AcuBench Report Composer is an interface for report design and layout that is very similar in appearance and behavior to the AcuBench Screen Designer. The interface includes a graphical design window onto which you can drag and drop report elements to lay out a report. As you add elements to the report, you can use a Property window to configure the appearance and behavior of the element and an Event Editor to tie code to the element.

Before you begin to design your report, you may want to complete the following preparatory steps:

1. If you are using relative, sequential, or Vision indexed files for your data, use the File Designer to create data layout files.
2. If you have created data layout files, define the corresponding data sets for your program.

   Note that these first two steps are not required. You can use the Report Composer to design reports regardless of what data format you are using. If, however, you are able to create data layout files and data sets to describe your data to AcuBench, the IDE provides tools to greatly simplify the report design process.
3. Configure the Report Composer interface, as described in this chapter.

This chapter discusses your options for customizing the tool to suit your needs. The next chapter provides detailed instructions for using the tool.

16.2 Customizing the Report Composer Interface

The Report Composer design window displays a report form similar in behavior and appearance to the screen form that appears in the Screen Designer. You add elements to this form to design your report.

By default, a grid of dots is displayed over each report form. This grid can be used to help you align elements of the report, creating precise columns and rows of printed output. The default size of each grid cell is 10 pixels by 10 pixels.

You can change the default grid behavior, as well as other report measuring behaviors, in the Tools/Options interface. To do this:

1. Open the Tools/Options interface, expand the **Report Writer** tree, and select **General**.
2. If desired, use the **Grid width** and **Grid height** entry fields to change the size of each grid cell. The measuring unit is pixels.
3. To force new report elements to align with grid cell boundaries, mark the **Snap to grid** check box. In a graphical report, you can still adjust the position of the control one pixel at a time using the arrow keys.
4. To simplify the process of aligning report elements in relation to one another, mark the **Auto alignment** check box. When this function is turned on,
   - if you position a report control underneath another control, and
   - the horizontal position of the two controls is within 5 pixels of one another, then
   - the new control's left boundary automatically aligns with the existing control's left boundary.
5. To control measuring behavior in the generated code, use the radio buttons in the Unit group. To measure sizes and distances in inches, select the **Inch** radio button. To measure in centimeters, select **Centimeter**.
6. In HTML reports, you have the option to choose whether embedded spaces in control titles or values are collapsed or preserved. To automatically collapse spaces, mark the **Collapse Spaces** check box. To preserve embedded spaces, unmark the box.

Note: Entry fields have a special Display Type property, used, among other things, to control spacing behavior. If the Collapse Spaces option is marked in the Tools/Options interface, the Display Type property *is* used. If the check box is not marked, values in the entry field are treated as though the "Keep Spaces" option were selected, regardless of the actual value set for the Display Type property.

---

7. If you have used the XFD tab of the File Designer to apply a NAME directive to any fields in your data files, you can choose to have the contents of the NAME directive appear in the Drag and Drop interface, rather than the actual field name. To enable this behavior, mark the **Apply XFD Names to Drag & Drop** check box.

   For more information about creating XFDs and the XFD NAME directive, see **Chapter 8, section 8.4.5, "Designing a Custom XFD."**

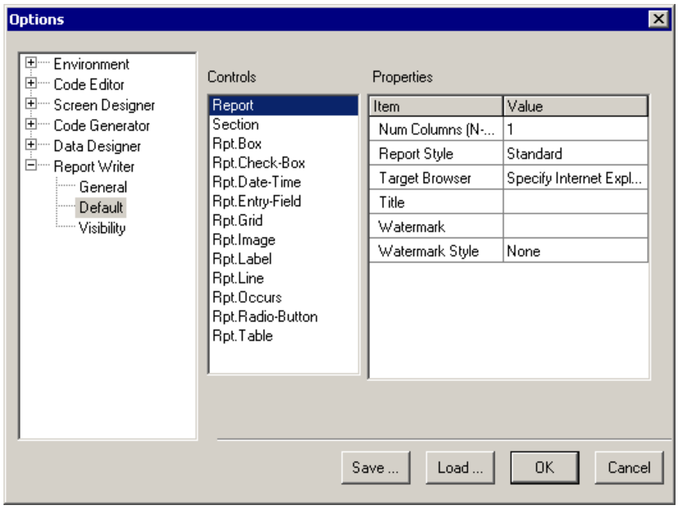
16.3 Establishing Report and Control Defaults

When you work in the AcuBench Report Composer, each element of the report (the report itself, each report section, and each report control) is associated with properties that appear in a Property window. You can determine the default setting for each of these properties and specify whether each property appears in the Property window. To enforce certain default settings, you could, for example, set the property default, then hide those properties in the Property window, causing the default property to always be used.

You can set the default properties and determine whether or not the property appears in the Property window through the Tools/Options interface.

To set default properties for report elements:

1. Open the Tools/Options interface, expand the **Report Writer** tree, and select **Default**.



Item	Value
Num Columns (N-...	1
Report Style	Standard
Target Browser	Specify Internet Expl...
Title	
Watermark	
Watermark Style	None

2. Select a report element from the Controls list along the left side of the interface. The properties corresponding to the selected element appear in the Properties list on the right.

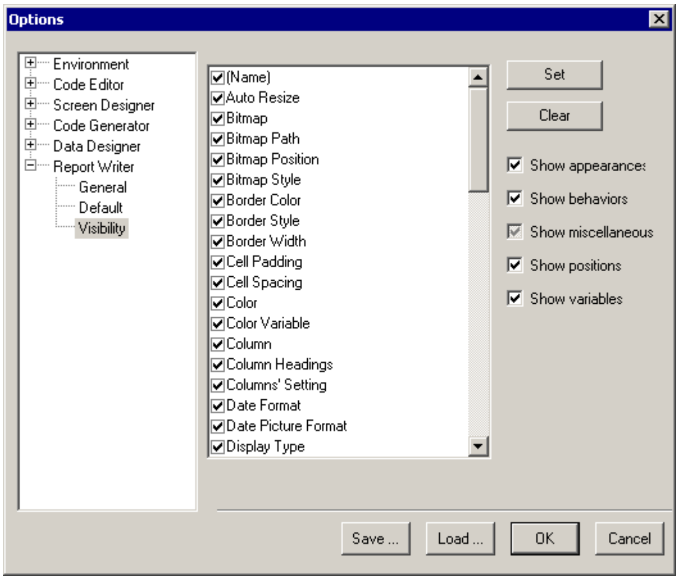
3. Scroll through the Properties list to view the available properties and their default settings. To change a setting, click in the Value column next to the property name and enter your change.

   In some cases, a drop-down list and/or push button may appear when you click in the Value column for a report element. In this case, you can select an item from a list or click the button to open a secondary interface to define the property setting.

   The result of any change to a report element's defaults is reflected in the Property window each time you create a new element of that type.

To determine which properties appear in the Property window when you work in the Report Composer interface, do the following:

1. In the Tools/Options interface, expand the **Report Writer** tree and select **Visibility**.



☑ (Name)
☑ Auto Resize
☑ Bitmap
☑ Bitmap Path
☑ Bitmap Position
☑ Bitmap Style
☑ Border Color
☑ Border Style
☑ Border Width
☑ Cell Padding
☑ Cell Spacing
☑ Color
☑ Color Variable
☑ Column
☑ Column Headings
☑ Columns' Setting
☑ Date Format
☑ Date Picture Format
☑ Display Type

☑ Show appearances
☑ Show behaviors
☑ Show miscellaneous
☑ Show positions
☑ Show variables

2. To remove an individual property from the Property window, remove the check box next to the property name in the large list box. To show the property in the Property window, mark the check box.

   You can also select one or more items in the list, then use the **Set** and **Clear** push buttons to add or remove properties from the Property window.

3. To remove an entire category of properties from the Property window, uncheck any of the **Show** check boxes (under the Set and Clear buttons). If you have removed a category from the Property window, you can add it back by marking the corresponding check box. Each property belongs to only one category.

16.4 Adding Report Templates

When you create a new report in AcuBench, you have the option to choose between two default templates (Blank Graphical and Blank Character). You can also create your own report templates and add them to the New Report dialog.

Using custom templates can simplify the process of report design and help you maintain consistency in your application output. You can, for example, create a template that includes a header or watermark containing your company logo, or that prints address or confidentiality information across the top or bottom of each page.

The process of creating a report template is described in **section 17.8**. Once you have the template file (".wtf"), you can use the Tools/Options interface to add the template to the New Report dialog.

To add a template to the New Report dialog:

1. Open the Tools/Options interface, expand the Environment tree, and select **Template**.

   More information about the Tools/Options/Environment/Template dialog box can be found in **section 4.3.2, "Template Options."**

2. In the "Customize template" section of the interface, select **Report** from the "Template for" drop-down box.
3. Click **Add** to open the Add New Template File dialog.
4. Enter a short, descriptive title in the "Template title" entry field, then click the browse (...) button next to the "Template file" entry field.
5. Navigate to the directory containing your ".wtf", select the file, and click **Open**.
6. Add a more verbose description of the template file in the Description field.
7. Click **OK** to save your changes, then click **OK** again to close the Tools/Options interface.

The next time you create a new report, your template appears in the New Report dialog, along with your description.

Tip: When you add templates to the New Report interface, a pointer is placed in the INI file, used to locate the ".wtf" file on disk. If you move the template file, AcuBench will no longer be able to locate the template and you will receive an error message. This means that if you are sharing a template among a team of developers, the template file should reside in a shared folder or be made part of your version control project.

16.5 Report Keyboard Shortcuts

Report-related keyboard shortcuts are configured in the Tools/Options interface. Expand the **Environment** tree and select **Keyboard**. The Report Composer keyboard shortcuts appear in the **Main** category. See **Chapter 4, section 4.3.6, "Keyboard Options,"** for more information about changing keyboard shortcuts.

17 Working with Reports

Key Topics

Introduction 17-2
Adding a Report 17-4
Formatting the Report 17-5
Adding Report Controls 17-12
Configuring Control Properties 17-15
Setting Print Conditions 17-16
Using Events to Populate a Report with Data 17-17
Creating Report Template Files 17-28
Sample Reports 17-29
Deploying in a Thin Client Environment 17-31

17.1 Introduction

When you want to create a report in AcuBench, a logical starting point is the definition of the data that you want processed and formatted into that report. After you have determined the information that your report should contain, you can design its appearance. The following general steps describe how to create a report in AcuBench:

1. Create or open the project and set up your work environment.
2. If you are working with data in sequential, relative, or Vision indexed files, create and refine your data layout (".dlt") in the File Designer.
3. If you have created one or more data layout files, create the corresponding data set(s) in the Data Set Designer.
4. Create a new report, then use the Report Composer interface to refine the report layout.
5. Make one-time modifications to the generated code.
6. Compile and run the program to generate the report.

This chapter discusses the types of reports that you can create with the Report Composer, the interfaces used to design a report, and the process of creating the report from start to finish.

17.1.1 Report Concepts: The Big Picture

What does it mean to design a report in the AcuBench Report Composer? At its core, a report generated by AcuBench functions like any other report written in COBOL:

1. A sequential read is performed on one or more data sources.
2. The program evaluates what to do next: print a footer, print a header, print a line of output, and so on.
3. The data is extracted. User-defined conditions determine whether data is moved to the print line and written to the print file.
4. The process repeats.

You can add layers of complexity, with headers and footers, nested breakpoints, multiple detail sections, and graphical report elements like grids and images. The basic report process, however, doesn't change.

17.1.2 Report Types

The AcuBench Report Composer can be used to create two basic types of reports: character and graphical.

A character report is a line sequential file with carriage return breaks at the end of each line, output to a standard print file. The print file can be stored on disk, viewed, or directed to a physical printer. Character reports include only text characters. They do not include graphical elements, such as bitmaps, grids, and tables. (They can, however, include simple lines, constructed from standard characters like hyphens, underscores, or asterisks.)

A graphical report is an HTML file, generated into the directory specified with the FILE_PREFIX configuration variable. The HTML file can be opened and viewed in a Web browser, or printed to a printer device directly from the COBOL program. Graphical reports can contain elements like bitmap and JPG files, as well as check boxes, radio buttons, grids, and tables.

In comparison with graphical reports, character reports:

- Include significantly fewer output lines, so the report can be produced more quickly
- Have lower memory requirements
- Do not require a browser for viewing

Graphical reports, on the other hand:

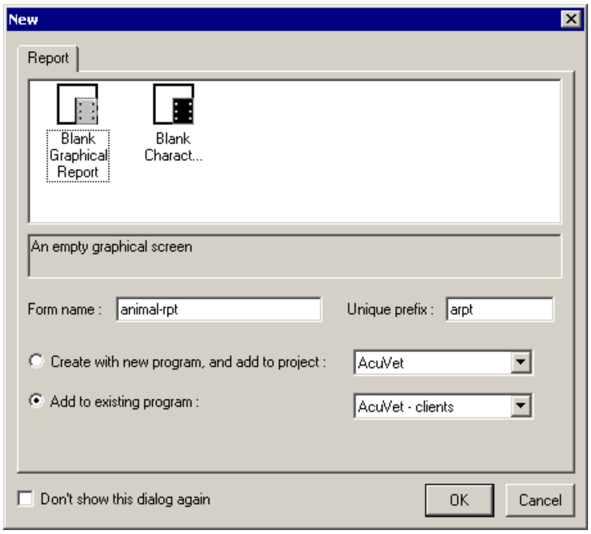
- Can include more complex elements, including grids and tables
- Provide more formatting options, including shading, colors, and images
- Can include HTML features like hypertext links, for added functionality

If you are printing very large reports, like a warehouse inventory report or a bank transaction report, you might choose a character report format to reduce overhead. If you are printing a client invoice or requisition form, on the other hand, you might choose a graphical report in order to allow for more complex formatting and for graphical elements like logos or bar codes.

17.2 Adding a Report

Once you have created an AcuBench program, the process of adding a new report is very straightforward:

1. In the Structure view, expand the node for the program to which you want to add a report.
2. Right-click the **Report** node for the program, then select **New Report**. The New Report interface is displayed.



An empty graphical screen

Form name : animal-rpt

Unique prefix : arpt

( ) Create with new program, and add to project : AcuVet

(•) Add to existing program : AcuVet - clients

[ ] Don't show this dialog again

3. Select a report template from the list at the top of the screen. The options you see depend on whether you have used the Tools/Options interface to add your own templates to the dialog (as described in **section 16.4**).

4. Enter a descriptive name for the new report in the **Form name** field, then modify the **Unique prefix** field, if necessary.

5. Verify that the report is being added to the correct project and program, then click **OK**.

   The new report opens in the Report Composer design interface, and an icon for the report is added to the program's Report node in the Structure view. You can open the report when needed by double-clicking on this icon.

If you have report templates that do not appear in the New Report interface, you can use those templates to create a report using the **Add Report** command. This is discussed in more detail in **section 17.8**.

17.3 Formatting the Report

When you create a new report using either of the default templates, the report opens with three default sections: a page header, a page footer, and a detail section. You can add or remove sections to determine the basic structure of your report, then configure report and section properties.

Note that you can add or remove sections at any point in the report creation process. Remember, however, that when you remove a section, any report controls that you have placed in that section and any properties that you have specified for that section are lost.

17.3.1 Adding and Removing Report Sections

An AcuBench-generated report can contain the following sections:

- Report Header
- Report Footer
- Page Header
- Page Footer

- Group Header (may include multiples)
- Group Footer (may include multiples)
- Detail section (may include multiples)

A *report header* is printed one time, at the very beginning of a report. It might contain information like the title of the report, the name of the person printing the report, and the date the report is being printed.

A *report footer* is printed at the very end of a report. It might include information like the time the report was completed, or a "completion" message to indicate that all necessary pages were printed.

A *page header* and *page footer* appear at the top and bottom of each page of the report. If you expect that the report you create will be printed to a device, having a page header and page footer can help to ensure the existence of proper page breaks. For this reason, even if you do not include any information in the page header or footer, it can still be useful to create the sections.

A *group* is a method of dividing up the data in your report according to breakpoints. You may want to preface a portion of your report with information about the items that follow, and/or conclude a portion of the report with summary or total information.

For example, if you are printing a daily report to summarize the transactions performed by five cashiers, the cashier ID might be a breakpoint. You could list the cashier ID first (*group header*), then, after listing the transactions, tally the daily total (*group footer*).

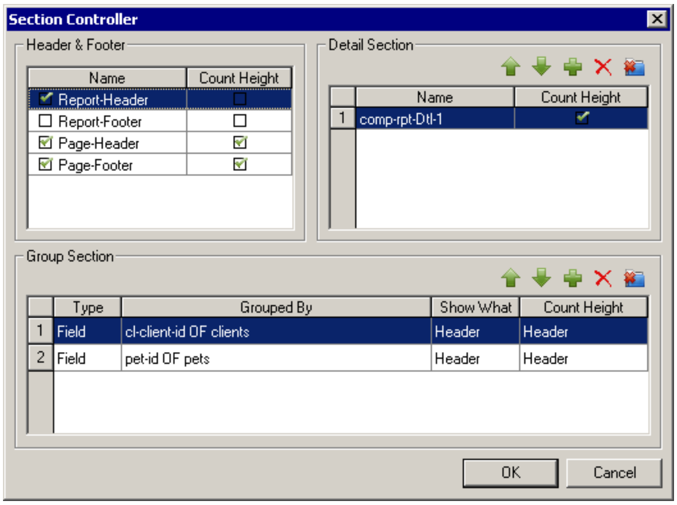
The *detail section* is the main body of your report. In a very simple report, you might read a record in a data file, print the record, then read the next record and repeat the process. In AcuBench terms, each of those printed records corresponds to a detail line.

In fact, a single detail line may include more than one physical row of output. You might have very long records, or construct a print line with data from multiple sources, or simply need a more readable layout that splits data items over multiple physical rows on the page. A detail section can contain as many physical rows as you need to print all of the data in the print line.

You also have the option to create multiple detail sections. When you have multiple detail sections, all of them correspond to one print line (or WRITE statement). In most cases, you can achieve the output results that you want with a single detail section. But the option to break the detail information for your report into different detail sections provides flexible shading options for readability and the option to set individual print conditions and before and after paragraphs for different segments of the same print line.

You can add or remove report sections with the Section Controller as follows:

1. Right-click on the report form in the Report Composer design window and select **Section Controller**. The Section Controller interface is displayed.



Name	Count Height
✓ Report-Header	☐
☐ Report-Footer	☐
✓ Page-Header	✓
✓ Page-Footer	✓

	Name	Count Height
1	comp-rpt-Dtl-1	✓

	Type	Grouped By	Show What	Count Height
1	Field	cl-client-id OF clients	Header	Header
2	Field	pet-id OF pets	Header	Header

2. Use the Header & Footer area to determine which header and footer sections, if any, will be included in your report. To add a section, mark the check box to the left of the section name.

   If you are adding a page header and page footer to help regulate page breaks, make sure that the **Count Height** check box to the right of the section name is also marked (with a red check mark). This prompts AcuBench to generate code to keep track of page height when positioning items on the page.

3. If you want to add additional detail sections to your report, use the Detail Section area of the Section Controller. (A report must have at least one detail section.)

   Click the **New** button (the third of the five buttons above the Detail Section list) to add a new detail section. Use the arrow keys to change the order of sections in the report. Use the **Delete** and **Delete All** buttons to remove sections.

   Remember that no matter how many detail sections you have, all of them correspond to one print line (WRITE statement), as described previously in this section.

4. To add breakpoints to your report, user the Group Section area of the Section Controller. To define a group, you must specify the field or records on which the report will break, then specify whether the report should include a group header (marking the start of the group), a group footer (marking the end of the group), or both. You cannot create a group section (set a breakpoint) without displaying either a group header or group footer.

   You can specify more than one group (breakpoint) in your report. If you have multiple groups, they are nested in the order that they appear in the Section Controller interface. So if you have created group headers and footers for two group items, for example, you will see the header for the first group, followed by the header for the second group, followed by the detail section. When all of the necessary detail lines have printed, you'll see the group footer for the second group, followed by the group footer for the first group.

   - Use the **Type** column to specify whether the report will break on a field or on records.

   - Use the **Grouped By** column to specify the field or records on which the report will break.

     If the report will break on a field, clicking the browse ("...") button in the Grouped By field opens a list of the data items defined in your DLTs, graphical Working-Storage, and definition files. If the report will break on records, clicking this button opens the Expression Builder.

- Use the **Show What** column to specify whether you want a group header, a group footer, or both.
- Use the **Count Height** column to cause AcuBench to generate additional positioning code to control where group headers and/or footers are printed in relation to their corresponding detail section(s).

5. When you are finished making changes, click **OK**. The report form is updated to include the sections that you have specified.

   To abandon your changes without updating your report, click **Cancel**.

17.3.2 Configuring Report and Section Properties

Each report and report section has configurable properties that you can modify through a Property window. By default, the Property window is displayed any time you open the Report Composer interface. If you have closed the Property window, you can open it again by selecting **Property Window** from the View menu, or by clicking the **Property Window** button on the Standard toolbar.

To configure properties of a report, expand the drop-down list at the top of the Property window, then scroll to the first entry (**Report:** ***reportname***) and select it. The Property window lists the properties associated with the entire report.

Properties that apply to the report as a whole include the following:

Property	**Description**
(Name)	The name used by AcuBench to refer to the report (by default, “Report1”). This name appears in a number of paragraph names in the generated report code.
Num Columns (N-Top)	If you are using an N-Top style report, this determines the number of columns used. The AcuBench default is “1”. This property is disabled for a Standard-style report.

Property	**Description**
Output File Name	The name of the file written to disk. By default, this is the name specified with the (Name) property with an ".html" extension.
Output File Variable	A variable used to hold the name of the file written to disk.
Paper Size	Click the browse ("...") button to open a dialog that allows you to select a paper size, set margins, and specify header and footer information. The default paper size is "Letter".
Report Style	There are two options: Standard or N-Top. If you plan to print your report to a device, select Standard. When you select N-Top, the report is configured to be viewed in a browser with a horizontal scroll bar. Because "N-Top" is a display orientation only, reports with this property are not meant to be printed.
Target Browser	Select from "Specify Internet Explorer" or "Standard HTML" options.
Title	The title entered here appears in the title bar when the report is viewed with a Web browser. It also appears in the first line of the print preview. Note that this title does not appear if you display the report with a Web browser control inside an ACUCOBOL-GT screen.
Watermark	Use this property to select a graphic as a background image for the report. Bitmap (".bmp" or ".dib"), JPG, and GIF formats are supported.
Watermark Style	Select from three options: None, Center, or Tile. If you select **None**, the default, the Watermark property is ignored and no image is printed.

When you work in the Report Composer interface, each report section is indicated by a gray bar, which appears above the section. This bar indicates the section type and name and may include additional information about the section. To change the properties for a section, click on this section bar. The name of the section appears in the drop-down box at the top of the Property window.

Section properties include the following:

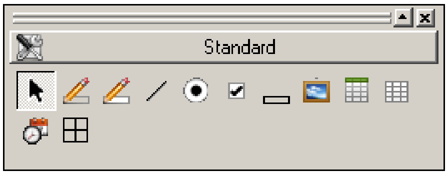
Property	**Description**
(Name)	The default section name is derived from the report name. You can change this name if desired.
Color	The default value for this property is 131329, foreground black on background white. Setting this property at the section level colors the background of the report and sets default colors for all report controls in the section.
Color Variable	Instead of setting a color value, you can specify a variable that will hold the color settings for the section.
Font	Setting the Font property at the section level sets the default font for all controls in the section.
Lines	This property represents the amount of space in a standard page that is reserved for printing a report section. This space is measured in either inches or centimeters, according to the Tools/Options settings that you have specified (see **section 16.2**).
Print After Page Footer	This property is available only for the report footer section. It determines whether the report footer prints before or after the page footer. The default value is "FALSE".
Print Before Page Header	This property is available only for the report header section. It determines whether the report header prints before or after the page header. The default value is "FALSE".

Property	**Description**
Print Condition	A print condition can be use to determine when a particular section is printed. AcuBench uses the condition that you enter here to construct an IF statement in the generated code. This means that your entry is something like: `user-name NOT = "chris"` You can type the print condition directly into the drop-down list in this property's Value cell (use **Ctrl+Enter** to move the cursor to a new line), or click the browse ("...") button to open the Expression Builder interface. The Expression Builder is discussed in **section 17.6**.
Size	Specify the width of the page in which this section is printing. As with the Lines property, the width is measured in either inches or centimeters, according to the Tools/Options settings that you have specified (see **section 16.2**).
Skip Page after Print	Select "TRUE" or "FALSE" to determine whether a page break is printed after the selected report section.
Skip Page before Print	Select "TRUE" or "FALSE" to determine whether a page break is printed before the selected report section.
Visible	Select "TRUE" or "FALSE" to determine whether the specified control or section is printed. Selecting "FALSE" for a group section prevents the controls in the group from printing, but does not inhibit detection of the breakpoint.
Visible Variable	Specify the name of the variable that will hold the Visible setting for the selected report section.
Zebra	This property applies only to the detail section. Enter a value between 1 and 16 to alternate different background colors for each physical row of the detail section. The specified background color alternates with white. The default value is "0".

17.4 Adding Report Controls

Report controls are elements that you can use to determine report layout and to format the data that will populate the report. Some of these controls affect the look and feel of the report, but are not directly related to report data (report line, box, and image). Other controls are concerned primarily with positioning and formatting data (report check box, date time, entry field, grid, occurs, radio button, and table). You choose the combination of controls that you need to create a useful and appropriate presentation for your data.

Report controls are listed in the Report Component Toolbox. If you are creating a graphical report, all of the controls in the Component Toolbox are enabled. If you are creating a character report, controls that include a graphical element appear disabled.



The Component Toolbox includes the following types of controls:

Report Label	Generally holds information describing another report element
Report Entry Field	A holder for data read into the report
Report Line	Adds a horizontal or vertical line to a report. Can be used to delineate columns or underscore totals, for example.
Report Radio Button	Used to indicate the value of a piece of report data when only one of a group of options can be selected
Report Check Box	Used with data that is either true or false. The box is checked when the corresponding data item has a value of “1” and unchecked when the item value is “0”.
Report Box	Draws a box, similar to the screen frame, around a report section or element. Does not include the grouping component of a frame control.

Report Image	Used to add one or more ".bmp" or ".jpg" images to a report
Report Grid	Used to provide a standard heading/column layout to data in your report. Can be used to add multiple records or database table rows to a single report detail section.
Report Occurs	Uses the OCCURS clause in a data table or array to create a line and column table layout for the data in the report
Report Date Time	Offers several formats for adding date or time information to the report
Report Table	Provides a method for structuring the output of a line of data. The table can have a vertical or horizontal orientation, and connects a header to a row of report data. The data can include literals, variables, and images.

17.4.1 Using the Report Component Toolbox

Add report controls to a report form in the same manner that you add a screen control to a screen form using the Screen Component Toolbox:

1. Click a control icon in the Component Toolbox.
2. Move the pointer to the report form. The pointer icon becomes a crosshairs.
3. Position the pointer on the form, then hold down the left mouse button and drag the mouse to specify the dimensions of the control. When you are finished, release the mouse button.

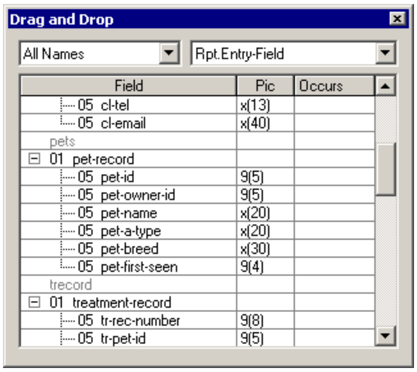
   The control appears on the report form.

17.4.2 Using Drag-and-Drop

As in the Screen Designer, if the data used to populate your report resides in graphical Working-Storage, data files described by a data layout file and data set, or AcuBench-recognized COPY files, you can use the Drag-and-Drop interface to add controls to your report. When you use this method to create controls, the controls are automatically associated with the selected data item(s).

To add a control to the report form using the Drag-and-Drop interface:

1. Right-click anywhere on the report form and select **Drag and Drop**.



Field	Pic	Occurs
05 cl-tel	x(13)	
05 cl-email	x(40)	
pets		
01 pet-record		
05 pet-id	9(5)	
05 pet-owner-id	9(5)	
05 pet-name	x(20)	
05 pet-a-type	x(20)	
05 pet-breed	x(30)	
05 pet-first-seen	9(4)	
trecord		
01 treatment-record		
05 tr-rec-number	9(8)	
05 tr-pet-id	9(5)	

2. Use the drop-down list on the left, if desired, to list the data items you want to use, then select a control type from the drop-down list on the right.

3. Highlight one or more data items in the “Field” list, then position the mouse over one of the selected items, hold down the left mouse button, and drag the selection to the report form.

   The specified control or set of controls is added to the report form.

17.4.3 Positioning and Alignment

In general, the same set of align functions is available for report controls as for screen controls. Use the Align menu and Align toolbar to size and position controls, either individually or in relation to one another.

When laying out report controls, it is important to remember that white (empty) space on the report for is preserved at runtime. This means that any extra padding that you leave above and below report controls is translated into empty space on the page when you print or view the completed report. As you finalize the layout of your report, remember to resize each report section to its content.

For detailed information about positioning and aligning controls, please see **Chapter 14, section 14.6, "Positioning and Aligning Controls."**

17.5 Configuring Control Properties

By configuring the controls that you have added to your report, you can determine exactly how data appears in the final report output. An entry field holding price information, for example, can be configured to apply an edited monetary format to a numeric field. A label containing header information can be given a bold typeface.

Although the principle of assigning properties to a report control is similar to that of configuring screen control properties, there are some significant differences:

- Each screen control property corresponds to a property entry in the Screen Section definition of the control. There is no parallel report control definition section; instead, property entries help to define how the procedure code used to print the control is generated.
- There tend to be fewer properties associated with a report control than with a screen control. Because a report entry field only displays data, rather than displaying and accepting information, it can't be enabled or disabled, for example, or have a minimum or maximum text setting. Likewise, since a report is formatted for a pre-selected page size, resize settings are unnecessary.
- Report controls can be associated with a print condition that determines whether or not it is printed during any given execution of the print loop. Similarly, report entry fields have a Print if Repeat property that lets you specify a general rule for how to handle consecutive records that contain the same data. You can either choose to always print the field (set Print as Repeat to "TRUE") or to not print duplicate fields (set Print as Repeat to "FALSE").

When you select a control on the report form, the Property window is updated to list the properties specific to that control. To change a property, click in the Value column. In some cases, you can type a new value in an entry field. In other cases, you can either select an option from a drop-down list or click a button ("...") to open a secondary interface used to configure the property value.

More information about the report controls and their properties can be found in **Chapter 18, "Chapter 18: The Report Controls and Property Reference."**

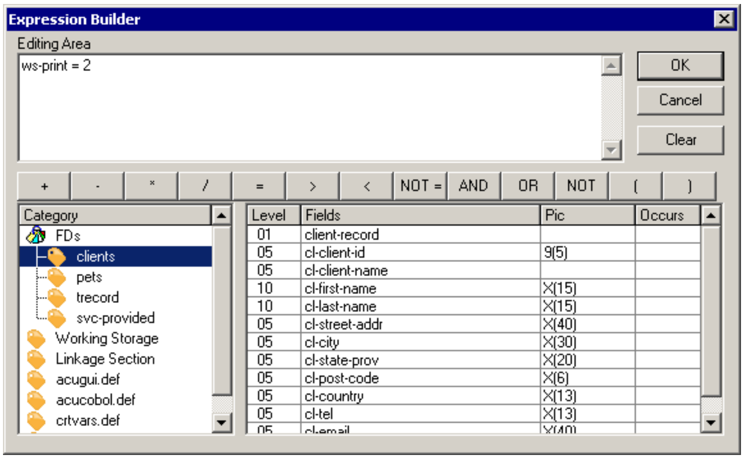
17.6 Setting Print Conditions

Print conditions allow you to establish rules that determine when a given report element is printed. You might want to create a sales report listing only those sales people who have not met their quota, for example, or create an inventory report for restocking purposes that only lists items of which there are five or fewer remaining. Likewise, in some circumstances, you might want to print an entry field control only if it contains a positive (or non-zero) value, or you might print either of two labels, depending on the value of some field in the current record.

To set a print condition for a given report section or control, do the following:

1. Select the report section or control.
2. In the property sheet, click in the Value column next to the **Print Condition** property.

3. Enter a simple condition (“my-counter = 1” or “zip-code NOT = 01003”) or click the browse (“...”) button to open the Expression Builder.



Editing Area

ws-print = 2

Category
- FDs
  - clients
  - pets
  - trecord
  - svc-provided
- Working Storage
- Linkage Section
- acugui.def
- acucobol.def
- crtvars.def

Level	Fields	Pic	Occurs
01	client-record		
05	cl-client-id	9(5)	
05	cl-client-name		
10	cl-first-name	X(15)	
10	cl-last-name	X(15)	
05	cl-street-addr	X(40)	
05	cl-city	X(30)	
05	cl-state-prov	X(20)	
05	cl-post-code	X(6)	
05	cl-country	X(13)	
05	cl-tel	X(13)	

The Expression Builder is a graphical interface that you can use to build print conditions. Click data elements and expression symbols to construct a condition statement. When AcuBench generates the code, this expression is added to an IF statement used to determine whether or not the given report element should be printed.

17.7 Using Events to Populate a Report with Data

Once you are satisfied with the design and layout of your report, your next step is to populate the report with data and possibly manipulate the output in some fashion. This is accomplished by writing code for events that perform these types of functions. These events are associated with the report itself and its elements, and are accessed from the Events tab. There are three types of events:

Before and After Print events: These type of paragraphs are tied specifically to individual elements of your report and the report itself. They allow you to code instructions for what you want to do before and after each report is generated.

BeforeDoPrint and AfterDoPrint events: These events are only associated with the report itself. They provide the code for performing READs and READ NEXTs, and initializing and resetting print loops. These paragraphs are what populates the report with data. Using these events from the Report Writer makes them part of the .psf file, which provides several advantages. These advantages, as well as coding samples are described in **Section 17.7.3**.

LoadGridInit and LoadGridNext Print events: These Grid control events provide the code for performing READs and READ NEXTs, and initializing and resetting print loops for grid controls. These paragraphs are what populates the grid with data. Using these events from the Report Writer makes them part of the .psf file, which provides several advantages. These advantages, as well as coding samples are described in **Section 17.7.4**.

Note: The BeforeDOPrint, AfterDoPrint, LoadGridInit and LoadGridNext events are new features of version 8.1 that provide a better way of coding reports. However, reports created with older versions of AcuBench are supported and can be regenerated without making any changes to your code. See **Section 17.7.5.5** for details.

The following sections describe the process of adding and coding these types of events.

17.7.1 Adding Report and Report Element Events

This section describes the process of adding and coding event paragraphs that populate a report with data and possible manipulate the data or the report in some fashion. The event paragraphs are summarized in **Section 17.7** and detailed in subsequent sections.

To add event paragraphs to a report, or report element such as graphical control or section, do the following:

1. Select the report element on the report form or the control on the form, or use the drop-down list at the top of the Property window to scroll to the desired element.
2. Select the Event tab of the Property window.

3. Click next to the paragraph type that you want to create.

   - If you are creating a new paragraph, click the browse ("...") button to specify a name for the paragraph and enter the Event Editor.
   - If you are pointing to an existing paragraph, select the paragraph name from the drop-down list.

4. If you are creating a new paragraph, the Add paragraph dialog appears, showing a default name for the paragraph (for example, "*report-element*-Before-Print"). You can either accept this name or enter a new name, then click **OK** to enter the Event Editor.

   To cancel without creating a paragraph, click **Cancel**. You still enter the Event Editor, but no new paragraph is created.

5. Enter the code for your new paragraph. For detailed paragraph descriptions, coding suggestions, and samples, see the sections on: **Before and After Event Paragraphs**, **BeforeDoPrint and AfterDoPrint Event Paragraphs**, **LoadGridInit and LoadGridNext Event Paragraphs**.

When you have finished creating and coding paragraphs for the report itself and all elements, generate the report and perform code modifications as described in **Section 17.7.5, "Generating Report Files and Code"**.

17.7.2 Before and After Event Paragraphs

Every report, report section, and report control can be assigned a Before Print and After Print paragraph. These events are used to perform initialization tasks before report generation and next or additional tasks after report generation.

Examples include:

- Open a relational database.
- Perform a calculation just before printing a total in an entry field.
- Move a numeric value from a COMP data type to a DISPLAY type (to show a monetary amount, for example).
- Update the color variable associated with an entry field (to show negative values in red, for example, in an expense tracking report).

- Update the page number in a page header or footer.
- Ask the user to confirm that report generation should start.

An After Print paragraph is your last chance to intervene before leaving the just-printed report element and moving on to the next one. You might use an After Print paragraph to:

- Add the data that's just been printed in a field to a running tally.
- Save a piece of data into Working-Storage before moving on to print the next item.
- Initialize a data item that has just been printed.

17.7.3 BeforeDoPrint and AfterDoPrint Event Paragraphs

Only the report itself contains BeforeDoPrint and AfterDoPrint events. You can provide paragraph names and code for these events using the steps described in **Section 17.7.1**. The code in these paragraphs perform READs and READ NEXTs, and initialize and reset print loops. They enable you to code instructions for what you want to do before and after each record is generated.

For example, you might define a BeforeDoPrint paragraph similar to this:

```
Myreport-BeforeDoPrint.
           move low-values to sales-key
           start sales key >= sales-key
           read sales-key next
                at end
                move 0 to Myreport-DOPRINTRTN-LOOP
           end-read
           .
```

And an AfterDoPrint paragraph that looked like this:

```
     Myreport-AfterDoPrint.
           read sales-key next
                at end
                move 0 to Myreport-DOPRINTRTN-LOOP
           end-read
           .
```

Note: Included with AcuBench are several sample reports such as "report1a" that utilize these event paragraphs. See **Section 17.9** for details on locating and using AcuBench sample reports.

To read data from a database with AcuSQL, the equivalent basic code might look like this:

```
PERFORM UNTIL sqlcode NOT = 0
   EXEC SQL
      FETCH my-cursor INTO :C-RECORD
   END-EXEC
   IF sqlcode = 0
      PERFORM Acu-RPT-report-DO-PRINT-RTN
   END-IF
END-PERFORM.
```

These events were created so that all of your report generation code will be part of the .psf (project structure file). This is has two major advantages:

1. All of your report generation code can be generated from your .psf file, which means your manual code will not be lost in the regeneration process.
2. You will only need to check in your .psf file into your version control system, as opposed to also having to check in the .cbl file.

17.7.4 LoadGridInit and LoadGridNext Event Paragraphs

Similar to the report element itself, the Grid control contains LoadGridInit and LoadGridNext print events. You name and code these paragraphs using the steps described in **Section 17.7.1**. The code in these paragraphs is used to initialize and READ data into your Grid control.

For example, you might define a LoadGridInit paragraph similar to this:

```
Myreport-LoadGridInit.
           move low-values to sales-key
           start sales key >= sales-key
           read sales-key next
                at end
                move 0 to myreport-mygrid-LOADGD-SW
           end-read
           .
```

And a LoadGridNext paragraph similar to this:

```
Myreport-LoadGridNext.
           read sales-key next
                at end
                move 0 to myreport-mygrid-LOADGD-SW
           end-read
           .
```

Note: Included with AcuBench is a sample report called “report1h” that utilize these event paragraphs. See **Section 17.9** for details on locating and using AcuBench sample reports.

The same advantages of using the events described in **Section 17.7.3, “BeforeDoPrint and AfterDoPrint Event Paragraphs”**, apply to these events as well.

17.7.5 Generating Report Files and Code

Once you have finished working in the Report Composer to create the code that populates your report, the next step is to generate your program. Varying degrees of code modification will then be necessary. The exact code changes required will depend on whether or not you defined and coded report and report element events, as described in **Section 17.7.1**.

When you generate the program, AcuBench places your event code in the .evt copybook. The rest of the report generation code is written to the .rpt copybook. (By default, this COPY file is placed in the Report folder in the File view.)

The following sections detail the code that is generated under several different circumstances including:

- Whether or not you defined BeforeDOPrint and AfterDoPrint events
- Whether or not you defined LoadGridInit and LoadGridNext events
- Existing reports imported from an earlier version of AcuBench

These sections also describes the code modifications that are necessary under each circumstance.

17.7.5.1 Reports Generated using BeforeDoPrint and AfterDoPrint events

If you defined BeforeDoPrint and AfterDoPrint events, as described in **Section 17.7.3**, the program's .cbl source file will contain the code to populate the data, and will look something like this:

```
REPORT-COMPOSER SECTION.
*{Bench}Myreport-masterprintpara
 Acu-RPT-Myreport-MASTER-PRINT-LOOP.

*Before do print paragraph
        PERFORM Myreport-BeforeDoPrint
     PERFORM UNTIL Myreport-DOPRINTRTN-LOOP = 0
         PERFORM ACU-RPT-Myreport-DO-PRINT-RTN
*After do print paragraph
         PERFORM Myreport-AfterDoPrint
  END-PERFORM
     .
*{Bench}end
```

Note that the Before and After do print paragraphs are inside AcuBench tags, which enables your code to become part of the AcuBench generated code.

With BeforeDoPrint and AfterDoPrint events, there is also a new variable generated for each report. Its definition is:

```
77 Myreport-DOPRINTRTN-LOOP   PIC 9 VALUE 0.
```

This variable is the switch that starts and stops the loop. To start the loop and thus the printing of data, you need to add a line of code similar to this one at the beginning of the PERFORM Myreport-BeforeDoPrint paragraph:

```
MOVE 1  TO Myreport-DOPRINTRTN-LOOP
```

17.7.5.2 Reports Generated Without BeforeDoPrint and AfterDoPrint Events

If you did not define and code BeforeDoPrint and AfterDoPrint events, as described in **Section 17.7.3**, your .cbl source file will look something like this:

```
*{Bench} report-masterprintpara
 Acu-RPT-report-MASTER-PRINT-LOOP.
  .
*{Bench}end
*
```

To add the code to read data into your report, you will need to manually code a DO-PRINT-RTN paragraph after the last AcuBench tag. For example:

```
PERFORM Acu-RPT-report-DO-PRINT-RTN.
```

Be aware that doing so will mean that this code will not be part of the .psf file. If the .cbl file is lost, the Read code is also lost. Also, it becomes necessary to perform version control on both the .cbl and .rpt files. For these reasons it is recommended defining BeforeDoPrint and AfterDoPrint events from within the Report Composer.

17.7.5.3 Grids Generated With LoadGridInit and LoadGridNext

If you defined LoadGrid events, the code to populate the grid is automatically added to the .cbl source file and looks something like this:

```
*{Bench}myreport:mygrid:-dogridrtn
       Acu-mygrid-TABBODY.
           PERFORM Acu-Initialize-mygrid
      *(Bench}initial grid load para
           PERFORM mygrid-LoadGridInit
           PERFORM UNTIL myreport-mygrid-LOADGD-SW = 0
                PERFORM Acu-mygrid-TabbodyPrintLoop
      *(Bench}next grid load para
              PERFORM mygrid-LoadGridNext
           END-PERFORM.

           PERFORM Acu-CLOSE-mygrid
           .
      *{Bench}end
      *
```

With LoadGrid events, there is also a new variable generated for each report. It’s definition is as follows:

```
77 myreport-mygrid-LOADGD-SW     PIC 9    VALUE 0.
```

This variable is the switch that starts and stops the loop. To start the loop and thus the printing of data, you need to add a line of code similar to this one at the beginning of the PERFORM mygrid-LoadGridInit paragraph:

```
MOVE 1  TO myreport-mygrid-LOADGD-SW
```

17.7.5.4 Grids Generated Without LoadGridInit and LoadGridNext Events

If you did not create LoadGrid events, AcuBench still adds lines to the .cbl source file that you can use to read multiple records into the control. The code for this paragraph, "Acu-*report-grid*-TABBODY", is generated between {Bench} tags. One PERFORM statement is isolated in such a way as to allow you to add code to populate the grid with data.

```
  *{Bench}Report1:Report-Grid:-dogridrtn
 Acu-Report-Grid-TABBODY.
     PERFORM Acu-Initialize-Report-Grid
*(Bench}initial grid load para

     PERFORM UNTIL Report-Report-Grid-LOADGD-SW = 0
        PERFORM Acu-Report-Grid-TabbodyPrintLoop
*(Bench}next grid load para

     END-PERFORM
     PERFORM Acu-CLOSE-Report-Grid
     .
*{Bench}end
```

For the same reasons as described with the report control, it is recommended that you define the code to populate the grid in the Report Composer via the Grid event tab.

17.7.5.5 Existing Reports Imported from Earlier Versions of AcuBench

Before and after do print events are a new feature in version 8.1. However, reports created with previous versions of AcuBench are still supported by AcuBench. When you regenerate the report with version 8.1 the only change made to your original code is the insertion of an additional line containing a period (.) within the bench tags. This will not cause any compilation issues. Your .cbl source file will look something like this:

```
*{Bench}report-masterprintpara
 Acu-RPT-report-MASTER-PRINT-LOOP.
      .
*{Bench}end
*
      PERFORM Acu-RPT-report-DO-PRINT-RTN.
```

If you want start coding using the new style you can go into the report composer and define the BeforeDoprint & AfterDoPrint events. You then can either manually delete the source outside the tags (starting with the `PERFORM Acu-RPT-`*`report`*`-DO-PRINT-RTN.`) or delete the source file and regenerate. Note however that once you remove your original code from the .cbl source file it cannot be brought back.

LoadGridInit and LoadGridNext events are also a new feature in version 8.1. However, reports with grids created with previous versions of AcuBench are still supported by AcuBench. When you regenerate the report with version 8.1 no changes are made to the AcuBench grid tags in your .cbl file.

If you want to start using these new grid events you will need to define and code them as described in **Section 17.7.1** and remove the previous tags by editing the .cbl source file or deleting the .cbl file itself. The new grid tags will only be generated if the old grid tags have been removed. Drawing a new grid control would insert the new tag for that control in the source, which makes it possible to have a mix of new and old style grid tags in the same source file.

17.7.6 Working with data from multiple sources

If you are retrieving data from several sources, the code gets more involved, but the central concept remains the same: you supply the code to retrieve the data needed to print one "line" of the report (regardless of whether that line is a record from a single data file or a collection of related data), then perform a generated routine to print that output in the format that you designed in the graphical Report Composer interface.

The code that follows demonstrates the process of populating a report with data from three indexed files. The report lists all of the clients of a veterinary clinic (recorded in a file called "clients"), all of the pets belonging to each client (recorded in a file called "pets"), and all of the treatments received by each pet (recorded in a file called “trecord”). In the interest of clarity, the file handling code has been simplified, leaving out INVALID KEY and AT END status checking.

The basic process retrieves data from the first file and uses that data to retrieve information from the second file. The data from the second file, in turn, is used to retrieve data from the third file. With all of the data in place, the print routine is executed to print a “line” (including group headers, group footers, and a detail section) of the report. The process then repeats until the entire report has been printed.

```
 create-history-report.
     MOVE LOW-VALUES TO cl-client-id.
     START clients KEY >= cl-client-id.
     PERFORM UNTIL ws-client-status = "10"
        READ clients NEXT RECORD
        IF ws-client-status NOT = "10" AND
           ws-client-status NOT = "02"
           PERFORM get-pets-for-client
        END-IF
     END-PERFORM.
*
 get-pets-for-client.
     MOVE cl-client-id TO pet-owner-id.
     START pets KEY = pet-owner-id.
     PERFORM UNTIL pet-owner-id NOT = cl-client-id
        READ pets NEXT RECORD
        IF pet-owner-id = cl-client-id
           PERFORM get-pet-treatment-history
        END-IF
     END-PERFORM.
*
 get-pet-treatment-history.
     MOVE pet-id TO tr-pet-id.
     START trecord KEY = tr-pet-id.
     PERFORM UNTIL tr-pet-id NOT = pet-id
        READ trecord NEXT RECORD
        IF tr-pet-id = pet-id
           PERFORM Acu-RPT-comp-rpt-DO-PRINT-RTN
        END-IF
     END-PERFORM.
```

17.7.7 Printing the Report

Before compiling and executing the program to create the report, you need to decide which forms of print output you intend to use. In the generated ".rpt" COPY file, AcuBench creates paragraphs that you can call to start the report process. These include:

- "Acu-*report*-PRINT-TOFILE" starts the report process and creates an output file on disk. If you are creating a graphical report, you can use this routine, then display the report on a graphical screen using the Web browser control. This can be significantly faster than using the preview function.
- The "Acu-*report*-PREVIEW" paragraph executes the code to create the report, then opens a preview window to display the report. Using this option requires you to have Internet Explorer, version 4.0 or later, installed on the machine running the report. Because of the overhead involved in communicating with Internet Explorer, this option is significantly slower than other options.
- The "Acu-*report*-DO-PRINT" paragraph executes the code to create the report, then sends the report to the default printer. You can use this paragraph as a template if you want to send the report to a different printer device.

Note: Both the PREVIEW and the DO-PRINT paragraphs require a DLL ("AcuBenchPrint.dll") that is automatically added to your AcuBench project when you create a report. If you are deploying in a thin client environment and using either of these routines to create and either preview or print the report, this DLL must be copied to the display host (client). For more information, see **section 17.10**.

To create your report, then, you need one line of code:

```
PERFORM Acu-report-PRINT-TOFILE.
```

Or:

```
PERFORM Acu-report-Preview.
```

Or:

```
PERFORM Acu-report-DO-PRINT.
```

Where you add this code depends on how you are invoking the report. If your program does nothing but create the report, add the PERFORM statement after the "PERFORM Acu-Initial-Routine" statement in the ".cbl" source file. If the report is created in response to a push button click or menu selection, add the PERFORM statement to the appropriate exception or link-to paragraph.

17.8 Creating Report Template Files

Any report that you create in the AcuBench Report Composer can be saved as a template file and used as the foundation for future reports. To create a report template:

1. Create and refine your report in the Report Composer.
2. Right-click anywhere on the report form and select **Generate WTF Document**. A Save As dialog opens.
3. Navigate to the directory in which you want to store the template file, enter a name for the file, and click **Save**.

Once you have create a report template, there are two ways that you can use the template to create a new report:

- Add the report to the New Report interface, as described in **Chapter 16, section 16.4, "Adding Report Templates."** When you use this option, each time that you create a new report, your template appears along with the two default templates in the New Report dialog.
- In the Structure view, right-click the Report node for the program in which you want to create a report, then select **Add Report**. In the Add Report to Program dialog, navigate to the folder containing your report template, select the template file, and click **Open**.

17.9 Sample Reports

Some sample reports have been created to help you get started with the Report Composer function. By default, these are placed in the "acugt/sample/reports" subdirectory of your Acucorp product installation directory. The samples include both character and graphical reports, and each report builds on the functions learned in the previous report. Together, the reports demonstrate the range of available report controls, from a simple report containing only report labels and entry fields to complex reports with tables and grids. The sample reports all use the same Vision indexed data files: "sales.dat" and "sales.vix". The data resides in the project's "Data" subdirectory.

The Reports sample project contains the following graphical report programs:

Program	**Description**
Report1a	This is a simple report that contains only entry fields and labels.
Report1b	This report is designed in landscape orientation and is ordered into columns. It includes labels, entry fields, and a date time control.
Report1c	This report includes group headers and a report occurs control.
Report1d	This report includes a report header and report footer, as well as report check box and radio button controls.
Report1e	This report demonstrates some uses for group footers.
Report1f	This report includes a report image control and a report table.
Report1g	This is an example of an N-Top report.
Report1h	This report demonstrates basic use of the grid control.
Report1i	This report provides a more advanced example of the report grid control, showing how to use the generated TABBODY code to load data into the control.

The Reports sample project also includes the following character report programs:

Program	**Description**
Report2a	This is a simple report that contains only entry fields and labels.
Report2b	This report is designed in landscape orientation and is ordered into columns. It includes labels, entry fields, and a date time control.
Report2c	This report includes group headers and a report occurs control.
Report2d	This report includes both page and report headers and footers, as well as nested group headers. It is presented in a landscape orientation.
Report2e	Building from Report2d, this report also includes nested group footers.

17.10 Deploying in a Thin Client Environment

AcuBench users running in a thin client environment with AcuConnect can print graphical reports from a UNIX application host. There are two different methods for accomplishing this:

1. Include a screen with a Web browser control in the deployed application, and use the control's built-in functionality to display and print HTML reports.
2. Use the AcuConnect "@[DISPLAY]:" notation in conjunction with the CODE_MAPPING configuration variable and the C$COPY library routine to access the preview and print functions included in the AcuBenchPrint.dll library file.

The sections that follow discuss each of these options and provide implementation information.

17.10.1 Using a Web Browser Control to Display and Print Reports

The ACUCOBOL-GT Web browser control, described in Chapter 5 of the *User Interface Programming* manual (Book 2 of the ACUCOBOL-GT documentation set), is a useful tool when you deploy a program that includes graphical reports. The Web browser control can parse and display both character and graphical reports, though its advantages are most clearly seen with graphical reports.

All you need to use this method to display and print reports are a screen containing a Web browser control and a program containing a report. The basic steps are as follows:

1. Design the screen and the report. Make sure that you know the output file name and path for the report.
2. Include code to perform the generated "Acu-*report*-Print-ToFile" paragraph.
3. Move the name and path of the generated report file to the value variable associated with the Web browser control.

   ```
   MOVE "c:\reporting\data\pets.html" TO ws-browser-url.
   ```

4. Add code to print the report. To allow the user to choose from various printers on the network, the code looks like this:

   ```
   MODIFY my-browser-control, PRINT=1
   ```

   Information about printing to a default printer or giving the user the option to perform page setup operations can be found in Chapter 5 of the *User Interface Programming* manual.

There are significant advantages to using this method to display and print reports:

- It's significantly faster than the "Acu-*report*-Preview" process.
- The Web browser control displays the report in full color and at 100% resolution.

- Any hyperlinks or other interactive aspects of the report are fully available to the user.
- You control the size of the Web browser control window.

17.10.2 Using AcuBenchPrint.dll to Display and Print Reports

When you invoke the "Acu-*report*-Preview" and "Acu-*report*-DO-PRINT" routines, AcuBench uses a library called "AcuBenchPrint.dll" to call the Internet Explorer functions needed to preview or print the report. This means that if you want to use AcuBench functions to preview or print a report in a thin client environment, you need to first copy the report and DLL to the report host, then perform a preview or print routine.

The general steps involved are as follows:

1. Place "AcuBenchPrint.dll" on the display host (either as part of your thin client installation or using C$COPY with "@[DISPLAY]" notation).

   This DLL should be placed in the same directory as the thin client executable ("acuthin.exe").
2. Use the CODE_MAPPING configuration variable to expose the function calls in the DLL.
3. Use the C$COPY library routine with "@[DISPLAY:]" notation to copy the HTML file from the application host to a location on the display host.
4. Add a line of code to reset the PRT-FULLFILENAME variable, which "AcuBenchPrint.dll" uses to locate the HTML file.

The following example demonstrates the code used to perform steps 3 and 4. In the sample program, the user pushes a button to create the report, which is then displayed with the preview function:

```
ReportPB-Link.
           IF IS-REMOTE
                 PERFORM ACU-REPORT1-THINPREVIEW
               ELSE
                 PERFORM ACU-REPORT1-PREVIEW
               END-IF.
     *
      ACU-REPORT1-THINPREVIEW.
            PERFORM Acu-Report1-PRINT-TOFILE
     *
            CALL "C$COPY" USING
                   "/home/data/Report1.html",
                   "@[DISPLAY]:C:\Localdata\Report1.html".
     *
            PERFORM Acu-Report1-PRINT-PARA
     *
            STRING
              "C:\Localdata\Report1.html", delimited by size,
              X"00", delimited by size,
              into PRT-FULLFILENAME.
     *
            SET ENVIRONMENT "DLL_CONVENTION" to "1"
            CALL "AcuBenchPrintDummy"
              ON EXCEPTION CALL "AcuBenchPrint.dll" END-CALL
            END-CALL
            CALL "AcuBenchPrintExecWBPreview" USING
               BY CONTENT PRT-FULLFILENAME,
               BY CONTENT PRINT-BROWSER-PARA,
               BY CONTENT ACU-PAPER-HEADER,
               BY CONTENT ACU-PAPER-FOOTER,
               BY CONTENT ACU-PAPER-SIZE,
               BY VALUE ACU-PAPER-ORIENTATIONINT,
               BY CONTENT ACU-PRINTER-NAME
            END-CALL
            .
```

The push button code uses the IS-REMOTE flag in the TERMINAL-ABILITIES structure to test whether the program is running locally or on a remote server. If it is running remotely, a paragraph called Acu-Report1-ThinPreview is executed.

The code in the Acu-Report1-ThinPreview paragraph starts by executing the AcuBench-generated PRINT-TOFILE paragraph. It then calls C$COPY to transfer the HTML file to the local machine. Next, it uses a STRING statement to reset the PRT-FULLFILENAME parameter to the location of the HTML file on the client machine. (Note that the PRT-FULLFILENAME string is terminated by low-values.) Finally, with code copied from the AcuBench-generated Acu-Report1-Preview paragraph (found in the ".rpt" COPY file), it calls the DLL and invokes the preview function.

With CODE_MAPPING set to "on", the configuration file can contain instructions for mapping these calls to "AcuBenchPrint.dll", which, in our sample, is located in the C:\Localdata directory on the client machine, as shown:

```
CODE_MAPPING ON
DLL_CONVENTION 1

ACUBENCHPRINT.DLL
@[DISPLAY]:C:\Localdata\AcubenchPrint.DLL
ACUBENCHPRINTDUMMY @[DISPLAY]:AcuBenchPrintDummy
ACUBENCHPRINTEXECWBPREVIEW
@[DISPLAY]:AcuBenchPrintExecWBPreview
```

The complete sample code for this example can be found in the Support section of the Micro Focus Web site. For detailed information about the "@[DISPLAY]:" notation and other thin client functions, refer to the *AcuConnect User's Guide*.

18 The Report Controls and Property Reference

Key Topics

Introduction 18-2
Common Report Control Properties 18-2
The Report Box 18-6
The Report Check Box 18-7
The Report Date Time 18-8
The Report Entry Field 18-10
The Report Grid 18-12
The Report Image 18-14
The Report Label 18-16
The Report Line 18-16
The Report Occurs 18-17
The Report Radio Button 18-19
The Report Table 18-20

18.1 Introduction

Whatever their other functions, both screens and reports provide a means of presenting data to the user. In either context, a control is a tool used to present a data item (or selection of data items), or to affect the appearance of data being presented.

Because of this similarity of purpose, there is a high degree of correspondence between screen controls and report controls. The bar control, for example, draws a vertical or horizontal line on a screen. The report line control, similarly, prints a vertical or horizontal line on a report. A screen grid control and report grid control both organize data into lines and columns with an optional heading row.

Report control properties, like their screen control counterpoints, can be divided in to *common* and *special* properties. Common properties apply to many or all control types, while special properties have meaning for one or few control types.

This chapter begins with an overview of common report control properties, then gives an overview of each report control type, describing the special properties that can be assigned to each.

18.2 Common Report Control Properties

Unless indicated otherwise, the control properties in the following list are common to all report controls. A property marked with an asterisk (*) can have its default value set in the Tools/Options/Report Writer/Default dialog. For a description of this interface, refer to **Chapter 16, section 16.3, "Establishing Report and Control Defaults."**.

(Name)

The default name is derived from your report name. For example, for Report1, the default name for the first label would be Report1-RwLa-1. You can change this name if desired.

Border Color

This property is not available for the check box, line, or radio button report controls.

A drop-down box allows for the selection of non-zero values from 1 to 16 that correspond to border colors. The default value for this property is “0”.

When the Border Style property is set to “Boxed”, the Border Color and Border Width properties combine to determine the presentation of the border around the report control.

Border Style

This property is not available for the check box, line, or radio button report controls.

A drop-down box allows for the selection of “Boxed” or “No Box”. For most controls, the default value for this property is “Boxed”. However, for the label and table report controls, the default value is “No Box”. Also, note that the “No Box” setting is ignored by the report box control.

When this property is set to “Boxed”, the Border Color and Border Width properties combine to determine the presentation of the border around the report control.

Border Width

This property is not available for the check box, line, or radio button report controls.

The Border Width property allows for the entry of an integer between 1 and 20, representing a width measurement in pixels. The default value for this property is “None”.

When the Border Style property is set to “Boxed”, the Border Color and Border Width properties combine to determine the presentation of the border around the report control.

Color*

The default value for this property is foreground black on background white.

The Value cell for this property has an ellipsis push button that opens the Color Setting dialog. Foreground and background colors can be set in this interface. The Color property setting of a report control overrides the Color setting of its parent section.

Color Variable*

The Value cell for this property has an ellipsis push button that opens the Select Variable interface. If a variable is selected in this dialog, subtle modifications to the generated code occur, such that it references the Color Variable instead of the Color value directly. This property has no initial default value.

Column

This property determines the location of the report control with respect to the left margin of the page. The unit of measure (inches or centimeters) is set in the Tools/Options/Report Writer/General interface.

Font*

This property is not available for the box, image, line, and occurs report controls.

A drop-down box displays several choices for font. The default value is "Default Font". The push button in the Value cell causes the Font dialog to appear.

The Font property setting for a report control overrides the Font property setting of its parent section.

Hyperlink

This property applies to report date time controls, entry fields, images, and labels. It is used to assign a hyperlink to the control when it is displayed in a Web browser. A hyperlink may reference either a local HTML file or a Web site.

To reference a local HTML file, click the ellipsis push button in the Value cell to display the Open dialog box. Navigate to the location of the local file and select it.

To reference a Web site, enter the entire address in the Value cell (e.g., http://www.microfocus.com). If you omit the "http://" prefix, AcuBench interprets the entry as a file name. In the latter case, the result is a "File not found" error message when you attempt to access the link.

This property has no default value.

Hyperlink Variable

This property applies to report date time controls, entry fields, images, and labels. The Value cell for this property has an ellipsis push button that opens the Select Variable interface. If a variable is selected in this dialog, subtle modifications to the generated code occur, such that it references the Hyperlink Variable instead of the Hyperlink value directly. This property has no default value.

Line

The Line property determines the location of the report control with respect to the beginning of the parent section. The unit of measure (inches or centimeters) is set in the Tools/Options/Report Writer/General interface.

Lines

This property is not available for the line report control.

The Lines property represents the height of the report control.

Note that the standard unit of measure used by Lines is set in the Tools/Options/Report Writer/General dialog. In this dialog, you can select Inch (the default) or Centimeter in the Unit frame.

Print Condition

A print condition can be used to make the printing of a report control conditional. Print conditions are discussed in detail in **Chapter 17, section 17.6, "Setting Print Conditions."**

Size

This property represents the width of the report control.

Note that the standard unit of measure used by Size is set in the Tools/Options/Report Writer/General dialog. In this interface, you can select Inch (the default) or Centimeter in the Unit frame.

Title

This property determines the text portion of the report check box, label, or radio button.

Title Variable

This property applies to the report check box, label, and radio button controls. The Value cell for this property has an ellipsis push button that opens the Select Variable interface. If a variable is selected in this dialog, subtle modifications to the generated code occur, such that it references the Title Variable, instead of the Title value directly. This property has no default value.

Visible*

A drop-down box allows the selection of "TRUE" or "FALSE". Selecting "FALSE" inhibits the printing of the report control. The default value is "1:TRUE".

Visible Variable*

The Value cell for this property has an ellipsis push button that opens the Select Variable interface. If a variable is selected in this dialog, subtle modifications to the generated code occur, such that it references the Visible Variable instead of the Visible value directly. This property has no initial default value.

18.3 The Report Box

The report box is similar to the ACUCOBOL-GT standard frame control. A report box, however, does not have a title, and does not perform any special grouping function for controls displayed inside the box. Like the report line, this report element is useful for highlighting some portion of the report text. You might, for example, place a box around the report title as a way of framing the text, or place a box around a group of entry fields or radio buttons in the detail section to make them stand out from the rest of the page.

The report box does not have special properties. All report box properties are described in **section 18.2, "Common Report Control Properties."**

18.4 The Report Check Box

The report check box is similar to the ACUCOBOL-GT standard check box control. It is used to display items that are either selected or unselected, true or false. If the variable associated with the control has a value of "1", the box is marked (contains a check mark). If the associated variable value is "0", the box appears unmarked (no check mark).

The report check box Property window contains the following special properties in addition to the properties described in **section 18.2, "Common Report Control Properties."**

Value

This property must be set to "0" or "1". A value of "0" (the default) indicates that the report check box is not selected in the report. A value of "1" indicates that the check box is selected.

The report check box is a stylistic display device. You need to code for the conditions that cause a particular check box to be selected. A logical place to perform this check is in a Before-Print paragraph for a report check box. For example, to manipulate a Sales Quota check box, the following code can be used in the Before-Print paragraph:

```
ADD heavy-equipment-sales TO supplies-sales
    GIVING total-salesperson-sales.
IF  total-salesperson-sales > sales-quota
    MOVE 1 to pr-quota
ELSE
    MOVE 0 TO pr-quota
END-IF.
```

When you display a report check box in Internet Explorer, the check box can be toggled by clicking on it.

For an example of this report control, refer to sample Report 1d in the sample reports area.

Value Variable

The Value cell for this property has an ellipsis push button that opens the Select Variable interface. If a variable is selected in this dialog, subtle modifications to the generated code occur, such that it references the Value Variable instead of the Value value directly. This property has no default value.

18.5 The Report Date Time

The report date time, like the screen date entry control, is used to present formatted date and time information. In a report, this information is likely most useful in a header or footer.

The report date time Property window contains the following control-specific properties in addition to the properties described in **section 18.2, "Common Report Control Properties."**

Date Format

This property determines how a date is displayed in the report date time. The default value for this property is "mm/dd/yyyy". A drop-down box allows the selection of the following formats:

yyyy/mm/dd
mm/dd/yyyy
dd/mm/yyyy
yy/mm/dd
mm/dd/yy
dd/mm/yy
yyyy/mm
mm/yyyy
yy/mm
mm/yy
mm/dd
dd/mm
None

Date Picture Format

This property defines the format in which dates are stored in Working-Storage. The default value for this property is "MMDDYYYY". A drop-down box allows the selection of the following formats:

YYYYMMDD
YYYYDDMM
MMDDYYYY
DDMMYYYY
YYMMDD
YYDDMM

MMDDYY
DDMMYY
YYYYMM
MMYYYY
YYMM
MMYY
MMDD
DDMM
None

Justification

A drop-down box allows the selection of “Center”, “Left”, “Right”, or “Unaligned”. The default value for this property is “Unaligned”.

Print If Repeat

A drop-down box allows the selection of “TRUE” or “FALSE”. The condition is applied when consecutive records contain the same data values. When this property is set to “TRUE” (the default), both data values print. When it is set to “FALSE”, the second (and subsequent) same data values do not print.

Time Format

This property determines how time is displayed in the report date time. The default value for this property is “None”. A drop-down box allows the selection of the following formats:

hh:mm:ss
hh:mm
hh
mm:ss
mm
ss
tt hh:mm:ss
tt hh:mm
tt hh
None

Note that “tt” is replaced by am or pm, “hh” by hours, “mm” by minutes, and “ss” by seconds.

Time Picture Format

This property defines how time is stored in Working-Storage. The default value for this property is "None". A drop-down box allows the selection of the following formats:

HHMMSS
HHMM
HH
MMSS
MM
SS
None

Value Picture

If you select an existing variable for Value Variable, Value Picture inherits the picture of the existing variable. If you create a new variable for Value Variable, you can create a picture clause for that variable here. If you leave Value Picture blank, a picture of X(30) is assigned to the new variable. This property has no default value.

Value Variable

The Value cell for this property has an ellipsis push button that opens the Select Variable interface. The Value Variable is not directly referenced by the generated code, but may be used by the user. This property has no default value.

18.6 The Report Entry Field

The report entry field corresponds to the ACUCOBOL-GT standard entry field control. Whereas a report label contains only literal values, a report entry field displays the value of a variable data item (its Value Variable property). When an entry field is tied to a file descriptor (FD) field, a simple READ NEXT statement can be used to update the field for printing. With formatted data or breakpoints, there may be additional coding considerations.

The report entry field Property window contains the following control-specific properties in addition to the properties described in **section 18.2, "Common Report Control Properties."**

Display Type

A drop-down box allows the selection of “Collapse”, “Keep Space”, and “Preformatted”. The default value for this property is “Collapse”.

The “Collapse” value allows only one space between two strings. With the “Keep Spaces” value, any amount of space between two strings is kept. A value of “Preformatted” means any format settings are kept. AcuBench generates the <PRE> and </PRE> HTML tags to maintain the text format.

Horizontal Spacing

The Value cell allows the entry of a digit with two decimal places. The default value for this property is “0.00”. Increasing this number enlarges the spaces between characters printed inside the report entry field.

Justification

A drop-down box allows the selection of “Center”, “Left”, “Right”, or “Unaligned” justification for the report entry field. The default value for this property is “Unaligned”.

Print If Repeat

A drop-down box allows the selection of “TRUE” or “FALSE”. The condition is applied when consecutive records contain the same data values. When this property is set to “TRUE”, both data values print. When it is set to “FALSE”, the second (and subsequent) same data values do not print. The default value is “TRUE”.

Value

This property can be a numeric or alphanumeric value. The value prints inside the report entry field. If a Value Variable is used, this numeric/alphanumeric is preserved as the initial Value of the Value Variable.

Value Picture

The Value Picture default value varies, depending on whether a Value property has been specified. If a Value has been specified, the Value Picture is derived from the Value. If no Value is specified, a default of X(30) is used.

Value Variable

The Value cell for this property has an ellipsis push button that opens the Select Variable interface. If a variable is selected in this dialog, subtle modifications to the generated code occur, such that it references the Value Variable instead of the Value value directly. This property has no default value.

18.7 The Report Grid

When you create a graphical report, you have the option to include a report grid, which is similar in appearance and function to the ACUCOBOL-GT standard grid control.

The report grid can be used when you want to display data in a standard column layout with a heading. Unlike the report table control, a report grid is always oriented with a heading row above one or more rows of data. The grid can be a single heading row with a single line of data (the grid is the result of one READ statement) or the grid can contain several rows of data (each grid line is the result of one READ statement).

Information about the code used to populate a grid with multiple rows of data can be found in **Chapter 17, section 17.7.5, "Generating Report Files and Code."**

The report grid Property window contains the following control-specific properties in addition to the properties described in **section 18.2, "Common Report Control Properties."**

Auto Resize

A drop-down box allows the selection of "TRUE" or "FALSE". With a setting of "TRUE" (the default), a report grid automatically expands to fit the data content size. The "FALSE" setting should be used with caution. When this property is set to "FALSE", the report grid uses the Lines property to determine its size, which may be inadequate to contain all your data.

Column Headings

A drop-down box allows the selection of “TRUE” or “FALSE”.

A setting of “TRUE” (the default) provides the user a place to enter a heading row for grid column titles.

A setting of “FALSE” removes the heading row. When Column Headings is set to “FALSE”, you can use the Page Header section of the report to provide the titles for the grid columns.

Columns’ Setting

The Value cell for this property has an ellipsis push button that opens the Columns' Setting dialog box. Click the Add push button in this interface to add a row with the following default settings:

Name: RwGd-1-Col-1

Pic: X(80)

Width: 1.00

Align: Unaligned

In the Name column, use the ellipsis push button to select a variable from graphical Working-Storage or from a data set. Set the Picture clause accordingly. In the Head column, type the column title. Width uses the unit of measure set in the Tools/Options/Report Writer/General interface. Use the Align column to set justification. Double-click in the More column to open a More dialog box, in which you can set Color, Font, and Hyperlink properties.

Repeat this procedure as needed for all of the fields you wish to have in your report grid.

Heading Color

The default setting for this property is foreground black on background gray. The Value cell for this property has an ellipsis push button that opens the Color Setting dialog. Foreground and background colors can be set in this interface.

Row Color Pattern

The Value cell for this property has an ellipsis push button that opens the Row Color Pattern dialog. The Add push button in this interface opens the Color Setting dialog, in which you can select background and foreground colors. You can observe your chosen pattern in the Preview area on the right of the Row Color Pattern interface.

Up and Down arrows permit you to rearrange color patterns you have defined, and a Remove button lets you delete color patterns.

Repeat this procedure to edit existing colors or to extend the Row Color Pattern.

Show Grid Line

A drop-down box allows the selection of "TRUE" or "FALSE". A setting of "TRUE" (the default) causes subsequent lines in a grid to be separated by a line. A setting of "FALSE" removes all grid lines in the grid.

18.8 The Report Image

Available only for graphical reports, the report image corresponds to the ACUCOBOL-GT standard bitmap control. You use the Open dialog to position a report image in the Report Composer window.

Note that in addition to placing one or more image controls on a report, you can also specify a Watermark property for the report (discussed in **section 17.3.2**) or display images in a report table control.

The report image Property window contains the following control-specific properties in addition to the properties described in **section 18.2, "Common Report Control Properties."**

Bitmap

The Value cell for this property has an ellipsis push button that displays the Open dialog box, from which you can navigate to "*.bmp" files. The Bitmap property records the name of the selected bitmap file. If the selected bitmap is not in a directory contained in the COPYPATH environment setting, then the full path to the bitmap is recorded here.

Bitmap Path

A drop-down box allows the selection of “Full Path”, “Dynamic Full Path”, and “User Defined”. These property selections affect how the bitmap file is located.

With the “Full Path” setting (the default), the browser uses the full path of the bitmap (as recorded when it was selected in the Report Composer) to locate the bitmap file.

When you set this property to “Dynamic Full Path” a call to the C$FULLNAME library routine is used to derive the full path of the bitmap file. The bitmap can be stored in any of the FILE-PREFIX directories named. The browser uses the full path of the bitmap to locate the bitmap file.

With the “User Defined” setting, the browser searches for the bitmap in the same directory as the HTML file.

Bitmap Position

A drop-down box allows the selection of “Center”, “Left Top”, “Left Bottom”, “Right Top”, and “Right Bottom”. Use these settings to position your bitmap in the report image control (when the Bitmap Style property is set to “Ratio”).

Bitmap Style

A drop-down box allows the selection of “Stretch” and “Ratio”. This property has no default value.

When Bitmap Style is set to “Ratio”, the bitmap’s height-to-width ratio is preserved inside the space allocated for the bitmap. If the bitmap display is smaller than the space allocated for it, the positioning function is similar to the justification of a text field, orienting to left, right, top, bottom, and center.

When Bitmap Style is set to “Stretch”, the bitmap height-to-width ratio is altered if necessary to fit the bitmap into the space allocated for it. Note that when Bitmap Style is set to “Stretch”, the Bitmap Position property setting is ignored.

Print If Repeat

A drop-down box allows the selection of "TRUE" or "FALSE". This condition is applied when consecutive records contain the same data values. When this property is set to "TRUE" (the default), both data values print. When it is set to "FALSE", the second (and subsequent) same data values do not print.

Value Variable

The Value cell for this property has an ellipsis push button that opens the Select Variable interface. The Value Variable is not directly referenced by the generated code, but may be used by the user.

18.9 The Report Label

The report label is similar to the ACUCOBOL-GT standard label control. The title of a report label, however, must be a literal value. You cannot associate a variable with a report label control. If you want to use a variable to display text in a report, use the report entry field control

The report label Property window contains the following control-specific properties in addition to the properties described in **section 18.2, "Common Report Control Properties."**

Justification

A drop-down box allows the selection of "Center", "Left", "Right", or "Unaligned" justification for the report label. The default value for this property is "Unaligned".

18.10 The Report Line

The report line is similar to the ACUCOBOL-GT bar control. This report element can be useful in a report's header and footer sections, for example, for underscoring column headings or totals. The report line Property window contains the following control-specific properties in addition to the properties described in **section 18.2, "Common Report Control Properties."**

Kind

A drop-down box allows the selection of "HORIZONTAL" and "VERTICAL". The default value is "HORIZONTAL".

Size (Lines)

This property represents the width of a horizontal report line or the height of a vertical report line.

Note that the standard unit of measure used by Size is set in the Tools/Options/Report Writer/General dialog. In this interface, you can select Inch (the default) or Centimeter in the Unit frame.

Width

The value of this property is a unit of measure in pixels. The default value is "1". For a horizontal line, this measure is the width of the line from top to bottom. For a vertical line, this measure is the width of the line from left to right.

18.11 The Report Occurs

The report occurs control lets you take advantage of the OCCURS clause to display report information in a table or array. It is available only for graphical reports. The report occurs Property window contains the following control-specific properties in addition to the properties described in **section 18.2, "Common Report Control Properties."**

Auto Resize

A drop-down box allows the selection of "TRUE" or "FALSE". A setting of "TRUE" (the default) automatically expands a report occurs to fit the data content size. The "FALSE" setting should be used with caution. When this property is set to "FALSE", the report occurs uses the Lines property to determine its size, which may be inadequate to contain all your data.

Column Headings

A drop-down box allows the selection of "TRUE" or "FALSE".

A setting of "TRUE" (the default) provides the user a place to enter a heading row for occurs column titles.

A setting of "FALSE" removes the heading row. When Column Headings is set to "FALSE", you can use the Page Header section of the report to provide the titles for the occurs columns.

Columns' Setting

The Value cell for this property has an ellipsis push button that opens the Columns' Setting dialog box.

In this interface, when you select the dominant occurs data item, the drop-down box at the top of the screen lists all the possible candidates in your graphical Working-Storage and data sets. To be a candidate, the data item must contain an OCCURS clause. The occurs displays in the top portion of the dialog box.

Level	Field Name	Picture	Usage	Occurs
03	Notes			8
05	Note-Date	X(10)		
05	Note-Initials	X(3)		
05	Note-Text	X(60)		

To transfer Note-Date, Note-Initials, and Note-Text from this table into the Occurs Columns table below, double-click them.

Name	Picture	Head	Width	Alignment	More
Note-Date	X(10)	Note-Date	1.00	Unaligned	
Note-Initials	X(3)	Note-Initials	1.00	Unaligned	
Note-Text	X(60)	Note-Text	1.00	Unaligned	

In the Head column, you can replace the listed name with a column title of your choosing. Width uses the unit of measure set in the Tools/Options/Report Writer/General interface. Use the Alignment column to set justification. Double-click in the More column to open a More dialog box in which you can set Color, Font, and Hyperlink properties.

Repeat this procedure as needed for all of the fields you wish to have in your report occurs.

Note that the Occurs Columns interface is similar to the Grid Columns interface, and also allows you to add and remove fields.

Row Color Pattern

The Value cell for this property has an ellipsis push button that opens the Row Color Pattern dialog. The Add push button in this interface opens the Color Setting dialog, in which you can select background and foreground colors. You can observe your chosen pattern in the Preview area on the right of the Row Color Pattern interface.

The Up and Down arrows permit you to rearrange color patterns you have defined, and a Remove button lets you delete color patterns.

Repeat this procedure to edit existing colors or to extend the Row Color Pattern.

Show Grid Line

A drop-down box allows the selection of "TRUE" or "FALSE". A setting of "TRUE" (the default) causes subsequent lines in a report occurs to be separated by a grid line. A setting of "FALSE" removes all grid lines in the report occurs.

18.12 The Report Radio Button

The report radio button is similar to the ACUCOBOL-GT standard radio button control. Like the ACUCOBOL-GT standard control, the report radio button can be assigned to a group and given a value within that group. The report radio button Property window contains the following control-specific properties in addition to the properties described in **section 18.2, "Common Report Control Properties."**

Group

The Group property setting is included for user documentation purposes. Radio buttons in the same group should be handled together, so that the selection or deselection of report radio buttons is coordinated.

Value

This property must be set to “0” or “1”. A value of “0” (the default) indicates that the report radio button is not selected in the report. A value of “1” indicates that the report radio button is selected in the report.

The report radio button is a stylistic display device. You need to code for the conditions that cause a particular radio button to be selected. A logical place to perform this check is in a Before-Print paragraph for a report radio button. The follow example shows one way to handle a group of radio buttons displaying credit card information:

```
INITIALIZE ws-credit.
EVALUATE card-type
   WHEN "M"
      MOVE 1 TO ws-mastercard
   WHEN "V"
      MOVE 1 TO ws-visa
   WHEN "A"
   MOVE 1 TO ws-amex
END-EVALUATE.
```

For another approach, refer to sample Report1d in the Reports sample project.

Value Picture

This property affects the picture clause of the generated value variable. It must be a numeric value. The default value for this property is “X(1)”.

Value Variable

The Value cell for this property has an ellipsis push button that opens the Select Variable interface. If a variable is selected in this dialog, subtle modifications to the generated code occur, such that it references the Value Variable instead of Value directly. This property has no default value.

18.13 The Report Table

In a graphical report, the report table allows you to format your detail section in a grid-like table. This control is not available in character reports.

At its simplest, a report table displays the result of a single READ statement (or equivalent) with corresponding header information. You can control whether the data is printed with a horizontal or vertical orientation. This can provide a useful way to align and group header and body information.

A report table can also be used to display images. Any cell in a table can be associated with an image file or image variable. The use of an image variable in a table is demonstrated in the Report1f sample program. In the sample, a group section has been established to organize the report data by state. A table displayed in the group header shows an image of the state flag associated with the selected state.

Additionally, a report table can be used to display multiple lines and columns of literal values. Multiline labels are not supported in reports, but a multiline table with no grid lines or border can be used to create the effect of a multiline label.

The report table Property window contains the following control-specific properties in addition to the properties described in **section 18.2, "Common Report Control Properties."**

Bitmap Path

A drop-down box allows the selection of "Full Path", "Dynamic Full Path", and "User Defined". These property selections affect how the bitmap file is located.

With the "Full Path" setting (the default), the browser uses the full path of the bitmap (as recorded when it was selected in the Report Composer) to locate the bitmap file.

When you set this property to "Dynamic Full Path", a call to the C$FULLNAME library routine is used to derive the full path of the bitmap file. The bitmap can be stored in any of the FILE-PREFIX directories named. The browser uses the full path of the bitmap to locate the bitmap file.

With the "User Defined" setting, the browser searches for the bitmap in the same directory as the HTML file.

Note that in the Table Setting dialog, the Cell Setting tab has a Type column, in which you can indicate that a table cell should contain an image. The Cell Setting tab then allows you to locate the bitmap file which should be displayed in that cell.

The Bitmap Path property applies to all image files so designated in the table.

Cell Padding

Cell padding adds blank space between the text in a cell and the horizontal grid lines above and below that text. The unit of measure for this property is pixels. Values of 1 through 5 are permitted, with a default value of “1”.

Cell Spacing

Cell spacing adds blank space between the vertical grid lines and horizontal grid lines in a table cell. The unit of measure for this property is pixels. Values of 1 through 5 are permitted, with a default value of “1”.

Merge Cell

Merge Cell lets you remove column dividers from a row.

The Value cell for this property has an ellipsis push button that opens the Merge Cell dialog. In this interface, click the left-most column of the adjacent columns you wish to merge, and drag your mouse across the columns you want to merge. Click Merge Cell and OK to return to the Report Composer window.

To restore the columns to the way they were, select the merged cell and click the Recover Cells push button.

Note that the Merge Cell operation removes any column titles or values from the selected columns. A Restore Cell operation restores the individual cells that you merged, but maintains only the title or value of the left-hand cell where you started the merge operation. Restore Cell is not an “undo” operation.

Also note that if you perform a Merge Cell operation on a table, you cannot subsequently perform the Table Rearrange function.

Show Grid Line

A drop-down box allows the selection of “TRUE” or “FALSE”. A setting of “TRUE” (the default) causes table cells to be separated by a grid line. A setting of “FALSE” removes all grid lines inside the table.

Table Rearrange

A table may be rearranged in any number of ways by row or by column and still retain an identical number of cells. For example, a 4 x 2 cell table may be rearranged as a 2 x 4, a 1 x 8, or an 8 x 1 cell table.

The Value cell for this property has an ellipsis push button that opens the Table Rearrange dialog box.

In a column measure rearrangement, the cells are selected from left to right, top to bottom, and reconfigured so that the same cells are displayed top to bottom, left to right.

In a row measure rearrangement, cells are selected from top to bottom, left to right, and reconfigured so that the same cells are displayed left to right, top to bottom.

Note that the Table Rearrange function is disabled if a Merge Cell operation has been performed.

Table Setting

The Value cell for this property has an ellipsis push button that opens the Table Setting dialog, with tabs for Column Setting, Row Setting, and Cell Setting.

The Column Setting tab lets you set column width, alignment, color, color variable, and font. In the Row Setting tab, you can set row height, alignment, color, color variable, and font. The Cell Setting tab lets you set each cell's type, data, pic, and alignment, along with elements in the More dialog.

Table Setting dialog's Column Setting tab

Column Width

This setting represents the width of the table column. The standard unit of measure is set in the Tools/Options/Report Writer/General interface.

Alignment

A drop-down box allows the selection of "Center", "Left", "Right", or "Unaligned". This field defaults to "Unaligned".

Color

The default value of “0” allows the column to inherit the color of the table. Double-click in the Color cell to activate an ellipsis push button that opens the Color Setting dialog. Foreground and background colors can be set in this interface.

Color Variable

Double-click the Value cell to activate an ellipsis push button that opens the Select Variable interface. If a variable is selected in this dialog, subtle modifications to the generated code occur, such that it references the Color Variable instead of the Color value directly. This field has no default value.

Font

A drop-down box displays several choices for font. The default value for this cell is “Default Font”. You can also double-click in the Font cell to activate an ellipsis push button that opens the Font dialog.

The column Font setting of a table overrides the Font property setting of its parent control.

Table Setting dialog’s Row Setting tab

Row Height

This cell represents the height of the table row. The standard unit of measure is set in the Tools/Options/Report Writer/General interface.

Alignment

A drop-down box allows the selection of “Middle”, “Top”, “Bottom”, or “Unaligned”. This field defaults to “Unaligned”.

Color

The default value of “0” allows the row to inherit the color of the table. Double-click the Color cell to activate an ellipsis push button that opens the Color Setting dialog. Foreground and background colors can be set in this interface.

Color Variable

Double-click the Value cell to activate an ellipsis push button that opens the Select Variable interface. If a variable is selected in this dialog, subtle modifications to the generated code occur, such that it references the Color Variable instead of the Color value directly. This field has no default value.

Font

A drop-down box displays several choices for font. The default value for this cell is “Default Font”. You can also double-click in the Font cell to activate an ellipsis push button that opens the Font dialog.

The row Font setting of a table overrides the Font property setting of its parent control.

Table Setting dialog’s Cell Setting tab

Cell

This field contains the coordinates of the table cell, presented in the format (X,Y). In this format, the cells in a table’s top row would have the coordinates (1, 1), (1, 2), (1, 3), etc. The cells of the table’s second row would have coordinates of (2, 1), (2, 2), (2, 3), etc.

Type

The selection of Type affects the behavior of the Data and Pic columns. A drop-down box allows the selection of “Text”, “Variable”, “Image”, and “Data Image”. This field defaults to “Text”.

Data

The contents of this field depend on the settings in the Type column. When the Type setting is “Text”, Data contains a character string, which is constant. When the Type setting is “Variable”, Data contains a variable selected from the Select Variable interface. When the Type setting is “Image”, Data contains a bitmap selected from the Open Files of Type *.bmp interface. When the Type setting is “Data Image”, Data contains a variable selected from the Select Variable interface

Pic

This field is used only when the Type setting is “Variable” or “Data Image”. It displays the picture clause of the variable represented in the adjacent Data cell.

Alignment

A drop-down box allows the selection of “Top-Left”, “Top-Center”, “Top-Right”, “Center-Left”, “Center”, Center-Right”, “Bottom-Left”, “Bottom-Center”, “Bottom-Right”, or “Unaligned”. This field defaults to “Unaligned”.

More

Double-click in the More cell to open the More dialog. You can set Color, Color Variable, Font, Hyperlink, and Hyperlink Variable properties in this interface.

19 Working with ACUCOBOL-GT Utilities

Key Topics

ACUCOBOL-GT Utilities 19-2
The Object File Utility 19-3
Using vio 19-7
Using logutil 19-10
Using vutil 19-11
Using acu4glfd 19-16
XML Support in AcuBench 19-17

19.1 ACUCOBOL-GT Utilities

AcuBench provides access to several of the ACUCOBOL-GT® tools and utilities through the Tools drop-down menu. Most of these utilities have a default command-line interface, but AcuBench provides an additional, graphical interface to each one. You access the graphical interface when you choose a utility from the Tools menu.

The tools and utilities accessed through the Tools menu include the following:

- **cblutil**, the object file utility
- **vio**, the file transfer utility
- **logutil**, the transaction log file utility
- **vutil**, the Vision file utility
- **xml2fd**, used to parse XML documents to produce FD and SELECT COPY files
- **AXDEFGEN**, the ActiveX definition generator
- **NETDEFGEN**, the .NET definition generator
- **acu4glfd**, used by the Acu4GL® family of interfaces to produce FD and SELECT COPY files for interoperating with relational database management systems

The first four utilities listed above (**cblutil**, **vio**, **logutil**, and **vutil**) are described in detail in Chapter 3 of the *ACUCOBOL-GT User's Guide* (Book 1 of the ACUCOBOL-GT manual set). The next three utilities (**AXDEFGEN**, **NETDEFGEN**, and **xml2fd**) are discussed in *A Guide to Interoperating with ACUCOBOL-GT.* The final utility, **acu4glfd**, is described in Chapter 6 in the *Acu4GL User's Guide*.

This chapter offers an overview of the utilities available from within AcuBench, describing any AcuBench-specific behaviors of the tool. It does not give detailed information about the general capabilities or uses of each utility. Please see the appropriate document (listed above) if you need more information about the functionality provided by any utility.

19.2 The Object File Utility

AcuBench offers a graphical interface to the ACUCOBOL-GT object file utility, **cblutil**. This utility can be used to assemble individual object files into object libraries, display information about object files, and translate portable object files (or libraries) into a native code format.

Note: This section describes only the AcuBench graphical interface to **cblutil**. For a complete description of **cblutil** functions, see Chapter 3 of the *ACUCOBOL-GT User's Guide*.

19.2.1 Object Libraries

An object library is a file that contains a group of one or more compiled ACUCOBOL-GT programs and its associated resource files (bitmaps, WAV files, and so on). Object libraries can simplify the distribution of an application by reducing the number of files involved. They can also help improve performance by reducing the number of directory operations performed by **runcbl** when it is loading object modules.

When you create an object library, the first module (compiled object) that you specify is called the primary module. When the library is loaded via a CALL statement or because it is the first program of the run unit, the primary module is the program that is loaded and run. Other modules in the library can be loaded by subsequent CALL statements.

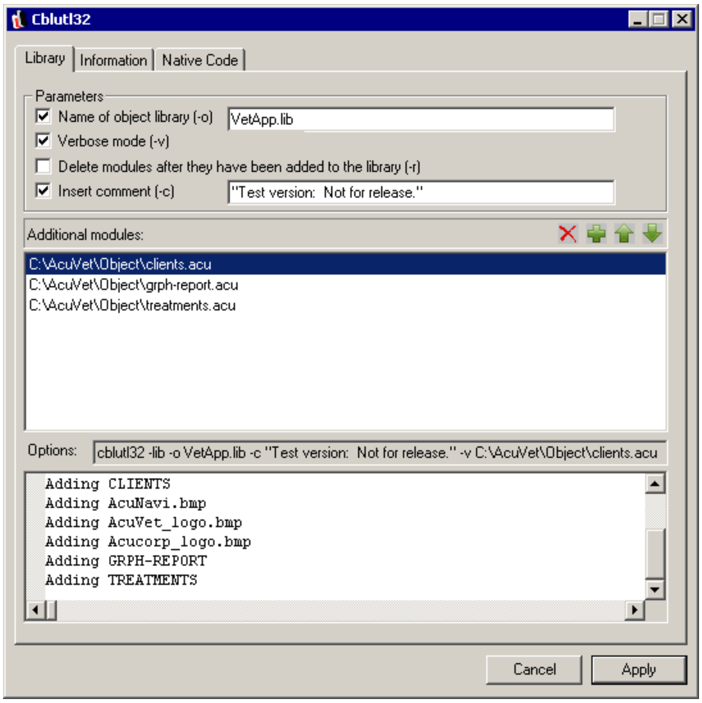
You should keep a couple of things in mind when you work with the **cblutil** object library utility:

1. A module that has the same name as a module already in the library will automatically replace the module of the same name. Therefore, before you build your library be careful to check that all modules have unique names.
2. We recommend that you place related object files in the same object library. For example, specify the main program of a run unit as the primary module and then add some or all of its subprograms.

19.2.2 Creating an Object Library

To access the object file utility from within AcuBench, open the Tools menu and select **Cblutl**. The Cblutl32 window opens with the Library tab selected.

The Library tab of the Cblutl interface is used to create an object library. The options contained in this interface correspond to the “cblutil -lib” command.



- [✓] Name of object library (-o): VetApp.lib
- [✓] Verbose mode (-v)
- [ ] Delete modules after they have been added to the library (-r)
- [✓] Insert comment (-c): "Test version: Not for release."

```
C:\AcuVet\Object\clients.acu
C:\AcuVet\Object\grph-report.acu
C:\AcuVet\Object\treatments.acu
```

Options: `cblutl32 -lib -o VetApp.lib -c "Test version:  Not for release." -v C:\AcuVet\Object\clients.acu`

```
Adding CLIENTS
Adding AcuNavi.bmp
Adding AcuVet_logo.bmp
Adding Acucorp_logo.bmp
Adding GRPH-REPORT
Adding TREATMENTS
```

To use this interface to create a library file:

1. Use the options in the Parameters section to build a **cblutil** command line.

   - Check the “Name of object library (-o)” check box and provide a name for the library you want to create in the entry field.

   - Specify whether you want the utility to operate silently or in verbose mode with the “Verbose mode (-v)” check box.

- Use the “Delete modules after they have been added to the library (-r)” check box to indicate how modules in the library are treated after the library is created.

- To include a comment in the compiled object, use the “Insert comment (-c)” field. Comments that include spaces must either be set off by quotation marks or include an escape character before each space.

As you make changes, the command line is displayed in the “Options” field, near the middle of the interface.

2. In the “Additional modules” area, use the **New** (“+”) button to add modules to the library. When you click this button, a browse (“...”) button appears in the field. You can use the browse button to navigate to your object files on disk, or type the path and file name in the field.

   To remove a module from the library, click the **Delete** button.

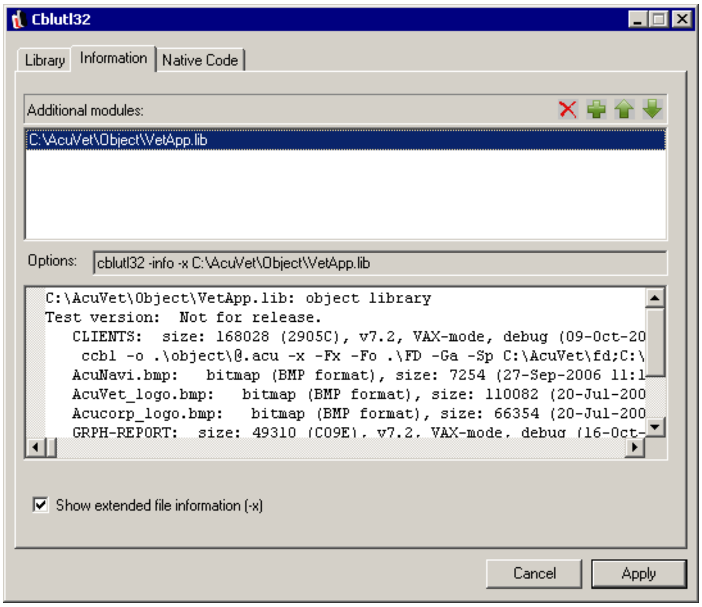
3. Use the **Up** and **Down** arrow buttons to re-order the modules as needed. The first item in the list will be the primary module at run time.

4. Click **Apply** to create your object library. Unless you have specified a different path in the “Name of object library (-o)” field, the library is created in the same directory as the primary module.

   Information about the object library that you have created appears in the box located below the “Options” entry field.

19.2.3 Retrieving Information About Objects

To display information about an object file (equivalent to the ACUCOBOL-GT "cblutil -info" command), select the **Information** tab of the Cblutl32 dialog.



C:\AcuVet\Object\VetApp.lib

Options: cblutl32 -info -x C:\AcuVet\Object\VetApp.lib

```
C:\AcuVet\Object\VetApp.lib: object library
Test version:  Not for release.
   CLIENTS:  size: 168028 (2905C), v7.2, VAX-mode, debug (09-Oct-20
    ccbl -o .\object\@.acu -x -Fx -Fo .\FD -Ga -Sp C:\AcuVet\fd;C:\
   AcuNavi.bmp:   bitmap (BMP format), size: 7254 (27-Sep-2006 11:1
   AcuVet_logo.bmp:   bitmap (BMP format), size: 110082 (20-Jul-200
   Acucorp_logo.bmp:   bitmap (BMP format), size: 66354 (20-Jul-200
   GRPH-REPORT:  size: 49310 (C09E), v7.2, VAX-mode, debug (16-Oct-
```

☑ Show extended file information (-x)

Here, you can specify one or more object files or object libraries and view information about how each was created and what resources are included in the object. As you make changes in the interface, the "Options" field displays the command line for the information function.

To retrieve information about an object file or object library:

1. Use the **New** ("+") button, then specify the name of the module(s) whose information you would like to view. You type a path and file name in the field or use the browse ("...") button to navigate to the file.

   Note that by default, the utility looks for files with a ".acu", ".cbx", or ".obj" extension. Expand the **Files of type** drop-down list in the Select Object File dialog to change the extension to ".lib" or other file extensions.

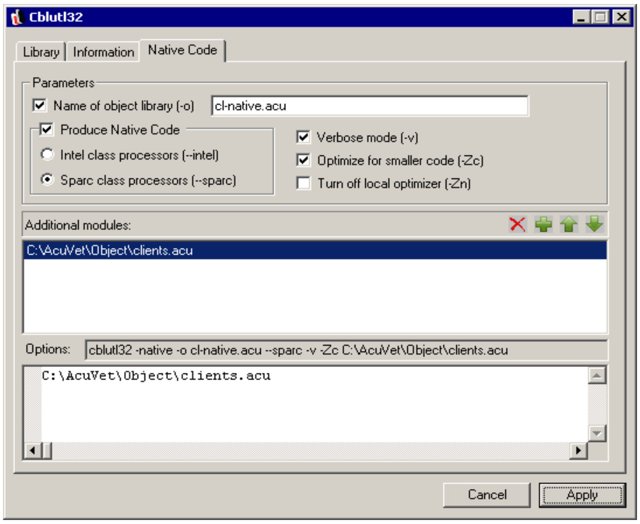
   To remove a module from the list, click **Delete** ("X").

2. To see the compiler options used to create each object in a library, as well as any comment information included in the object or library, mark the **Show extended file information (-x)** check box.

3. Click **Apply** to activate the information function. Information from the header of the selected object file appears in the box located below the "Options" field.

19.2.4 Generating Native Code

To translate ACUCOBOL-GT portable object modules into native code object modules, select the **Native Code** tab of the Cblutl32 window. This is equivalent to invoking the "cblutil -native" command.



[✓] Name of object library (-o): cl-native.acu
[✓] Produce Native Code
Intel class processors (--intel) [✓ Sparc class processors (--sparc)]
[✓] Verbose mode (-v)
[✓] Optimize for smaller code (-Zc)
[ ] Turn off local optimizer (-Zn)

Additional modules: C:\AcuVet\Object\clients.acu

Options: cblutl32 -native -o cl-native.acu --sparc -v -Zc C:\AcuVet\Object\clients.acu

```
C:\AcuVet\Object\clients.acu
```

To create a native object or library:

1. Use the Parameters section of the interface to specify the type of native code that you want to create.

   - Provide a name for your object library in the entry field to the right of the "Name of object library (-o)" check box.

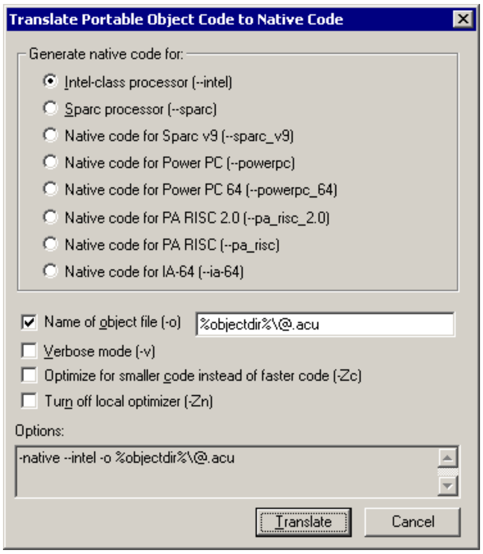
- Specify the processor type for which you want your object optimized.
- Select optimization options and indicate whether or not the utility should operate in verbose mode using the check boxes to the right of the “Produce Native Code” area.

2. Specify the module(s) that you want translated into native code in the “Additional Modules” entry box. As in the other Cblutl32 interface, use the **New**, **Delete**, and arrow buttons to arrange modules.

   The “Options” entry field displays the command line for the native code function.

3. Click **Apply** to generate native code object modules.

Note that you can also transform portable objects to native code from the File view of the Workspace window. Right-click an object in the Object folder and select **Native Code**.



Translate Portable Object Code to Native Code

Generate native code for:

- (●) Intel-class processor (--intel)
- ( ) Sparc processor (--sparc)
- ( ) Native code for Sparc v9 (--sparc_v9)
- ( ) Native code for Power PC (--powerpc)
- ( ) Native code for Power PC 64 (--powerpc_64)
- ( ) Native code for PA RISC 2.0 (--pa_risc_2.0)
- ( ) Native code for PA RISC (--pa_risc)
- ( ) Native code for IA-64 (--ia-64)

- [✓] Name of object file (-o) %objectdir%\@.acu
- [ ] Verbose mode (-v)
- [ ] Optimize for smaller code instead of faster code (-Zc)
- [ ] Turn off local optimizer (-Zn)

Options:

```
-native --intel -o %objectdir%\@.acu
```

In the Translate Portable Object Code to Native Code interface, select a processor type and a series of commands to pass to **cblutil**. Click **Translate** to create the native object file.

19.3 Using vio

vio is a file transfer and archive utility that allows you to create file archives and access archive content. **vio** is well suited for moving files to different operating systems because it is available on most platforms supported by ACUCOBOL-GT and because it automatically adjusts for certain machine-dependent aspects, such as byte-swapping. It also easily handles multiple volumes.

vio runs in two modes: input mode and output mode. Use output mode to create an archive. Use input mode to read and extract from an archive. Each mode has a distinct set of options. For a complete description of **vio**, including its options, see section 3.4 of the *ACUCOBOL-GT User's Guide*.

The vio Interface

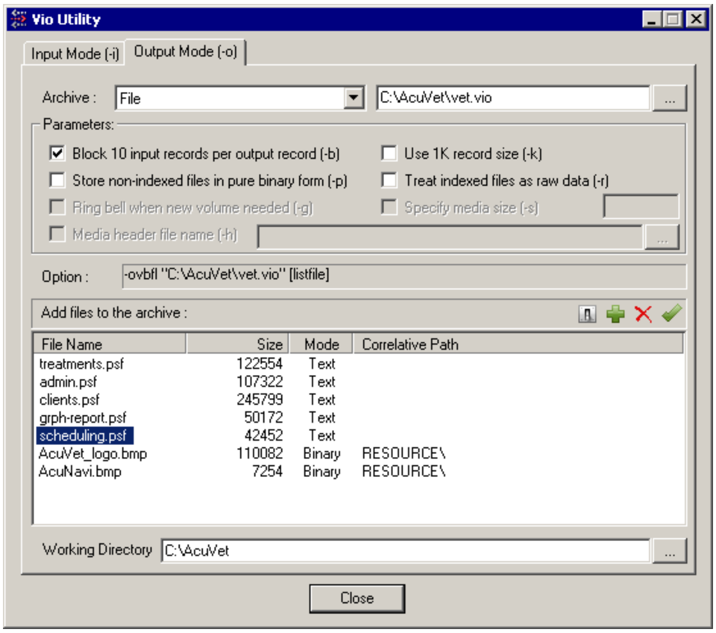
To launch **vio**, select the **Vio** command from the Tools drop-down menu. AcuBench displays the Vio Utility dialog.

The Vio Utility dialog includes two tabs, Input and Output. Select the mode that you want to use by clicking on the corresponding tab. You can switch between tabs at any time.

Note: Use caution when specifying full path names. Some operating systems do not interpret or translate full paths properly. It is preferable, in most cases, to use relative path names when transferring files to a different operating system.

19.3.1 Output Mode

When you want to create a file archive, click on the **vio** Output tab.



Archive : File   C:\AcuVet\vet.vio

Parameters:

- [✓] Block 10 input records per output record (-b)
- [ ] Use 1K record size (-k)
- [ ] Store non-indexed files in pure binary form (-p)
- [ ] Treat indexed files as raw data (-r)
- [ ] Ring bell when new volume needed (-g)
- [ ] Specify media size (-s)
- [ ] Media header file name (-h)

Option : -ovbfl "C:\AcuVet\vet.vio" [listfile]

Add files to the archive :

File Name	Size	Mode	Correlative Path
treatments.psf	122554	Text	
admin.psf	107322	Text	
clients.psf	245799	Text	
grph-report.psf	50172	Text	
scheduling.psf	42452	Text	
AcuVet_logo.bmp	110082	Binary	RESOURCE\
AcuNavi.bmp	7254	Binary	RESOURCE\

Working Directory C:\AcuVet

Create a **vio** archive using the following steps:

1. From the “Archive” entry field drop-down list, select the target storage device. Selecting “File” causes the archive to be placed on the system hard disk.

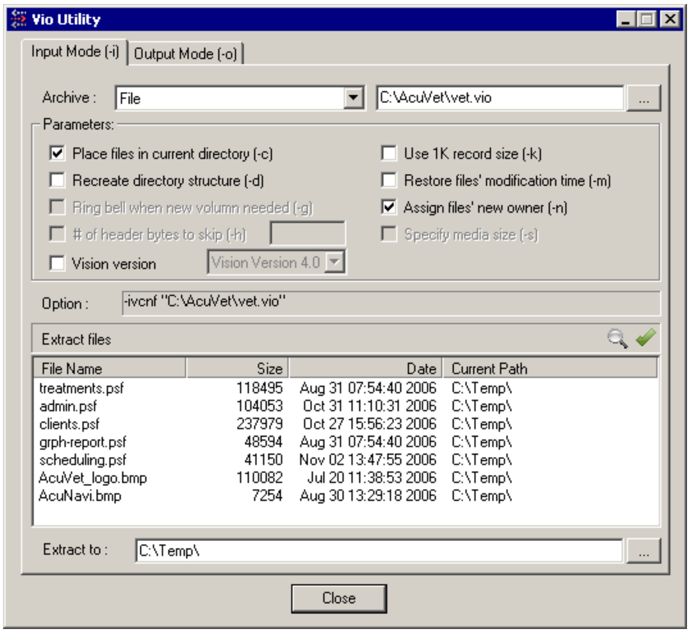
2. In the entry field to the right of the “Archive” drop-down list, enter the path and name of the archive, or click the ellipsis button to navigate to an existing file.

3. In the Parameters section, select the options that you want. The “Option” field displays the command line that reflects the options chosen. You cannot directly edit the contents of this field. For a complete description of all **vio** options, see section 3.4 of the *ACUCOBOL-GT User’s Guide*.

4. Build the list of files to be archived in the Collect files area. Click the **Add files** button in this area to build your list. The Add Files dialog allows you to browse the file system to locate the file(s) you want to add. Use the **Delete** button to remove a selected file. Select the **Include Subdirectories** check box to include all files and subdirectories in the current directory. Repeat this procedure until all the files you want have been added.
5. To toggle the file type of any file on the list, select the file from the list and click the **Toggle mode** button. The current designated file type for any file on the list is indicated under the “Mode” heading.
6. To build the archive, click the **Start collect** button.
7. When you are finished, click **Close** to exit **vio**.

19.3.2 Input Mode

When you want to read and extract from a file archive, click on the **vio** Input tab.



Archive : File C:\AcuVet\vet.vio

Parameters:

- [✓] Place files in current directory (-c)
- [ ] Recreate directory structure (-d)
- [ ] Ring bell when new volumn needed (-g)
- [ ] # of header bytes to skip (-h)
- [ ] Vision version — Vision Version 4.0
- [ ] Use 1K record size (-k)
- [ ] Restore files' modification time (-m)
- [✓] Assign files' new owner (-n)
- [ ] Specify media size (-s)

Option : -ivcnf "C:\AcuVet\vet.vio"

Extract files

File Name	Size	Date	Current Path
treatments.psf	118495	Aug 31 07:54:40 2006	C:\Temp\
admin.psf	104053	Oct 31 11:10:31 2006	C:\Temp\
clients.psf	237979	Oct 27 15:56:23 2006	C:\Temp\
grph-report.psf	48594	Aug 31 07:54:40 2006	C:\Temp\
scheduling.psf	41150	Nov 02 13:47:55 2006	C:\Temp\
AcuVet_logo.bmp	110082	Jul 20 11:38:53 2006	C:\Temp\
AcuNavi.bmp	7254	Aug 30 13:29:18 2006	C:\Temp\

Extract to : C:\Temp\

You can read and extract a **vio** archive as follows:

1. Select the device that hosts the archive file from the "Archive" drop-down list. Select **File** if the archive is located on a system disk.

2. In the entry field to the right of the "Archive" drop-down list, enter the path and name of the archive file, or click the ellipsis button to navigate to the archive file.

3. In the Parameters section, select the desired options. The "Option" field displays the command line that reflects the options chosen. You cannot directly edit the contents of this field. For a complete description of all **vio** options, see section 3.4 of the *ACUCOBOL-GT User's Guide*.

4. Read the archive. Click **View** to read the archive file and display its contents.

5. Extract the archive. In the "Extract to" entry field at the bottom of the **vio** window, enter the path of the target directory, or click the ellipsis button to navigate to a target directory.

   To extract the contents of the archive to the target directory, click the **Start Extract** button at the far right of the "Extract files" label. The contents of the archive are placed in the target directory.

Click **Close** to exit **vio**.

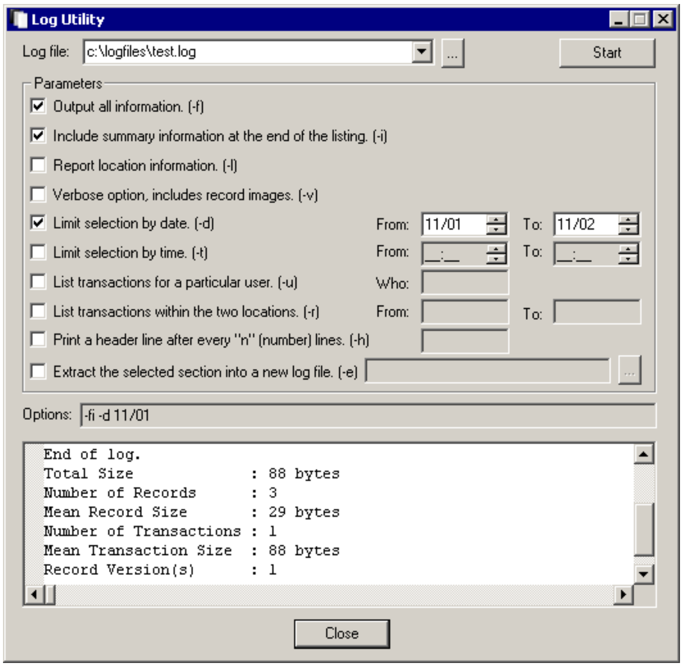
19.4 Using logutil

ACUCOBOL-GT provides basic transaction management facilities, including extended COBOL transaction syntax and transaction logging capabilities. The utility program **logutil** is used to examine and edit ACUCOBOL-GT transaction log files built specifically for the Vision file system.

This section provides a description of the graphical interface to **logutil** provided by AcuBench. For a complete description of ACUCOBOL-GT transaction management facilities, see section 5.1 of the *ACUCOBOL-GT User's Guide*. For a complete description of the command line interface to **logutil**, as well as the report format, see section 3.6 of the *ACUCOBOL-GT User's Guide*.

The graphical interface to logutil

To launch **logutil** from within AcuBench, select the **Logutil** command from the Tools drop-down menu. AcuBench displays the Log Utility dialog.



Log file: c:\logfiles\test.log

- [✓] Output all information. (-f)
- [✓] Include summary information at the end of the listing. (-i)
- [ ] Report location information. (-l)
- [ ] Verbose option, includes record images. (-v)
- [✓] Limit selection by date. (-d) From: 11/01 To: 11/02
- [ ] Limit selection by time. (-t) From: _:_ To: _:_
- [ ] List transactions for a particular user. (-u) Who:
- [ ] List transactions within the two locations. (-r) From: To:
- [ ] Print a header line after every "n" (number) lines. (-h)
- [ ] Extract the selected section into a new log file. (-e)

Options: -fi -d 11/01

```
End of log.
Total Size              : 88 bytes
Number of Records       : 3
Mean Record Size        : 29 bytes
Number of Transactions  : 1
Mean Transaction Size   : 88 bytes
Record Version(s)       : 1
```

You can generate a report from a transaction log file as follows:

1. Enter the name of the log file in the “Log file” field. You can either type the name directly into the field or use the browse (“...”) button to navigate to and select the file.

2. In the Parameters section, select your options. The “Option” field displays the command line that reflects the options chosen. You cannot directly edit the contents of this field. For a complete description of all **logutil** options, see section 3.6 of the *ACUCOBOL-GT User’s Guide*.

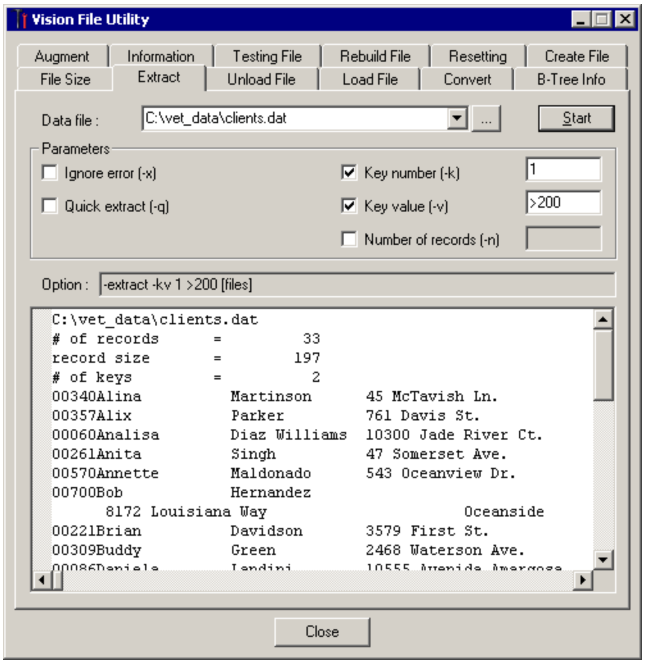
3. To start the report, click **Start**. The results are displayed in the lower portion of the window.

   To interrupt a report that is in progress, click the **Stop** button (the Start button becomes a Stop button after the search begins).

4. When you are finished, click **Close** to exit the utility.

19.5 Using vutil

The ACUCOBOL-GT Vision file utility, **vutil**, is used with Vision indexed files to extract file structure information, extract records, rebuild corrupted files, retrieve deleted records, and more. When you work in AcuBench, you can interact with this utility through a graphical interface accessed from the Tools menu. This section describes that graphical interface to **vutil**.



Data file : C:\vet_data\clients.dat

Parameters

- [ ] Ignore error (-x)
- [ ] Quick extract (-q)
- [✓] Key number (-k) 1
- [✓] Key value (-v) >200
- [ ] Number of records (-n)

Option : -extract -kv 1 >200 [files]

```
C:\vet_data\clients.dat
# of records      =        33
record size       =       197
# of keys         =         2
00340Alina          Martinson      45 McTavish Ln.
00357Alix           Parker         761 Davis St.
00060Analisa        Diaz Williams  10300 Jade River Ct.
00261Anita          Singh          47 Somerset Ave.
00570Annette        Maldonado      543 Oceanview Dr.
00700Bob            Hernandez
      8172 Louisiana Way                      Oceanside
00221Brian          Davidson       3579 First St.
00309Buddy          Green          2468 Waterson Ave.
00086Daniela        Landini        10555 Avenida Amargosa
```

For a complete description of the command line utility and its functions, see section 3.3 of the *ACUCOBOL-GT User's Guide*.

Note: vutil does not make use of the runtime configuration file. Settings made in a runtime configuration file do not affect **vutil** in any way.

The basic steps involved in invoking any **vutil** function are as follows:

1. To open **vutil**, select the **Vision File Utility** command from the Tools drop-down menu.

The graphical Vision File Utility dialog includes several tabs, each of which provides access to a different **vutil** function. To select a function, simply click the corresponding tab.

2. Because **vutil** operates on Vision indexed files, each **vutil** function requires that you specify the name of a Vision file. You can either type the name of the file in the "Data file" (or "Source file" or "File name") field, or click the browse ("...") button to navigate to the file.

   As long as the **vutil** dialog is open, the name of each file specified is added to a drop-down list associated with the file name field. You can easily select a previously referenced file from this list.

3. If you have chosen a function that can take parameters, use the Parameters section to build your **vutil** command line.

   An "Option" field displays the command line that reflects the options chosen. Note that you cannot directly edit the contents of this field.

4. Click **Start** to perform the operation that you have specified. The results of the operation appear in the area near the bottom of the dialog.

   To stop a **vutil** function that is in progress, click the **Stop** button (the Start button becomes a Stop button when a **vutil** function is in progress). If you attempt to select another **vutil** tab while the current action is in progress, the message "Vision File Utility is running. Please wait." is displayed.

5. To close the **vutil** interface, stop or allow the current action to complete and click **Close**.

19.5.1 Increasing the Maximum File Record Size

The **vutil** augment function is accessed on the Augment tab. The augment function lets you increase the maximum record size of a Vision file. This option is useful if you want to add fields to a record without having to rebuild the entire data file.

19.5.2 Examining File Information

The **vutil** information function is accessed on the Information tab. The information function extracts basic information from the specified Vision file. All extracted information is displayed in the bottom half of the tab.

19.5.3 Testing File Integrity

The **vutil** test file function is accessed on the Testing File tab. The test file function checks a Vision file for integrity. Integrity is examined on such points as whether a file is corrupt and whether it has a non-zero user count. Testing for file integrity should be one of the first actions taken in assessing data file problems.

19.5.4 Rebuilding Files

The **vutil** rebuild file function is accessed on the Rebuild File tab. The rebuild file function recreates or rebuilds the specified indexed file. You should rebuild a file that has become corrupt, or one that contains a large number of deleted or lost records that you want to remove from the file. The **vutil** rebuild function has many options that allow you to control how the rebuilt file is created.

Note that the “-m”, “-s”, and “-r” options are not available in the graphical interface to **vutil**. To invoke these options, use the command-line interface to the utility.

19.5.5 Resetting User Counts

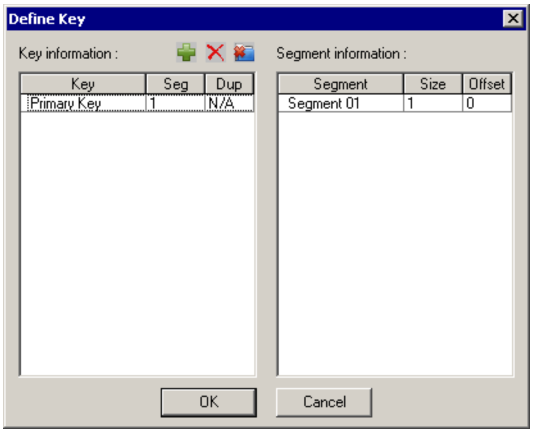
The **vutil** reset function is accessed on the Resetting tab. This function allows you to reset the user count of the specified files to zero. The reset function eliminates the need to rebuild the file when the file has a non-zero user count and you are certain that the file is not corrupted.

19.5.6 Creating Empty Files

The **vutil** function that allows you to create a new, empty file is accessed on the Create File tab. The create file function is equivalent to performing an OPEN OUTPUT on a file in COBOL.

When you use the Create File tab of the Vision File Utility interface, there are a few points to keep in mind:

- If you specify the name of an existing file when you invoke this function, that file will be overwritten.
- By default, **vutil** creates the new file in Vision Version 5 format. If you want another format, use the "Vision file format" check box and drop-down list to make an alternate selection.
- You can use the **Define Key Information** push button to open a special interface used to create, delete, and define keys for the new file. Use the buttons at the top of the screen to create or delete keys. Double-click in any key field to change its value.



Key	Seg	Dup
Primary Key	1	N/A

Segment	Size	Offset
Segment 01	1	0

For a complete description of the **vutil** create function, see section 3.3 of the *ACUCOBOL-GT User's Guide*. For general information about Vision files, see section 6.1.3 of the *ACUCOBOL-GT User's Guide*.

19.5.7 File Size Summary

Use the File Size tab to access the **vutil** file size function, which reports summary disk usage information for the Vision file. Information returned includes total file size, the number of allocated records in use, the number of empty records, and the percentage of allocated records in use.

19.5.8 Extracting Records

The **vutil** extract record function is accessed on the Extract tab. The extract function allows you to display records stored in the specified file. The target file cannot be encrypted. If you attempt to use the extract function to view a record in an encrypted file, the message “encrypted, extract not allowed” is displayed.

After you enter a file name, specify extract parameters, and start the record extraction, the records matching your parameters are displayed in the field at the bottom of the Vision File Utility window.

19.5.9 Unloading to Other File Types

The **vutil** unload function is accessed on the Unload File tab. The unload function creates a binary sequential or line sequential file from the specified Vision indexed file.

19.5.10 Loading a File

The **vutil** load function is accessed on the Load File tab. The load file function creates an indexed file from a binary sequential file, relative file, or line sequential file.

19.5.11 Converting Indexed Files

The **vutil** convert function is accessed on the Convert tab. The convert function allows you to convert an indexed data file created by C-ISAM, Micro Focus, or RM/COBOL-85 into a Vision indexed file. This is useful when you are moving data from a C-ISAM, Micro Focus, or RM/COBOL-85 application to an ACUCOBOL-GT application.

Note: Because the Vision file replaces the original C-ISAM, Micro Focus, or RM/COBOL-85 file, be sure that you have a backup copy of the original file.

The convert function has a number of limitations. It does not convert files that have:

- record or block sizes greater than 32 KB
- more than 120 keys
- individual keys larger than 250 bytes
- non-ASCII characters or collating sequence (RM/COBOL-85)
- split keys (RM/COBOL-85)
- a single key with more than 16 segments (Vision Version 4) or more than six segments (Vision Version 2 or 3) (C-ISAM)
- a primary key that allows duplicates (C-ISAM)

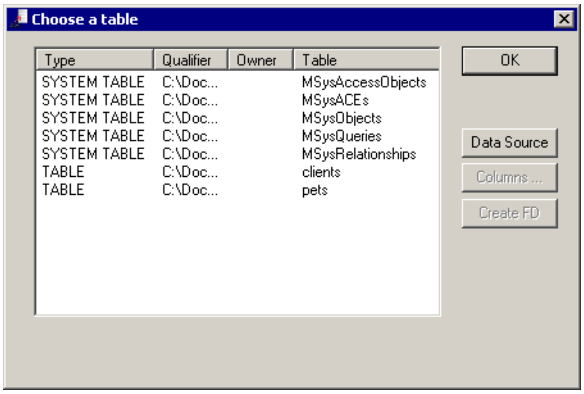
Note: When converting files you must have plenty of available disk space to accommodate the differences in file size between the original file type and the converted file type. You must also allow **vutil** space to copy the file during the conversion process.

19.5.12 B-Tree Listing

The **vutil** tree function is accessed on the B-Tree Info tab. The tree function outputs B-tree structure information for the specified file. It is used primarily as an aid in debugging suspected Vision file problems.

19.6 Using acu4glfd

A Windows utility called **acu4glfd** helps you create file descriptions (".fd") and SELECTS (".sl") for use with ***extend's*** Acu4GL interfaces. It can also determine a unique index, which it uses as the primary key.



Type	Qualifier	Owner	Table
SYSTEM TABLE	C:\Doc...		MSysAccessObjects
SYSTEM TABLE	C:\Doc...		MSysACEs
SYSTEM TABLE	C:\Doc...		MSysObjects
SYSTEM TABLE	C:\Doc...		MSysQueries
SYSTEM TABLE	C:\Doc...		MSysRelationships
TABLE	C:\Doc...		clients
TABLE	C:\Doc...		pets

Because this utility uses ODBC technology to obtain table information in order to list table fields, you need to have Acu4GL installed for the runtime to access the table.

You access this utility's interface via the Tools/acu4glfd command. You need to choose and connect to a data source. When the tables contained in the data source are displayed, users can direct the utility to create FDs and SELECTs.

More information about this utility can be found in section 6.2 in the *Acu4GL User's Guide*.

19.7 XML Support in AcuBench

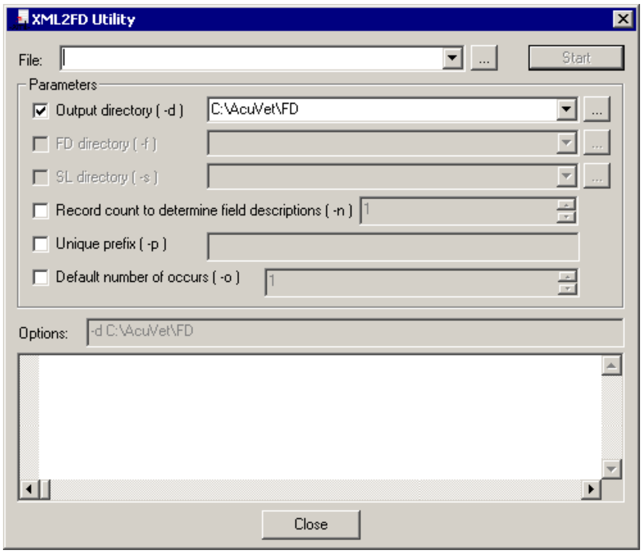
The ACUCOBOL-GT runtime has a file system interface for XML files, called AcuXML, that seamlessly converts XML data to COBOL data and vice versa. Information about working with XML data can be found in section 8.2, "Working with XML Data," in *A Guide to Interoperating with ACUCOBOL-GT*.

When you make use of the AcuXML interface, you can use the **xml2fd** utility to derive an FD and SELECT statement from an XML file. When a file of type XML is selected, the import process must pass through the **xml2fd** utility. This utility's interface helps to guide you through the import of the XML file to an FD file, which is written into a project's ".fd" folder and imported into the AcuBench graphical File Designer.

xml2fd also returns a list of the lines where it's guessing about the data type. This prints to a "stderr" file and can be viewed in the Output window. The graphical File Designer is displayed, along with Output window information. When you click on a line in the Output window, the cursor is positioned at the line in the File Designer where the "guessing" occurred. You can then change the Picture clause, if desired, prior to regenerating the code.

To use **xml2fd** from within AcuBench, do the following:

1. Select **Xml2FD** from the Tools menu, or right-click a project node in the Data view and select **XML2FD**.



XML2FD Utility

File: | ... | Start

Parameters

[✓] Output directory (-d) C:\AcuVet\FD ...
[ ] FD directory (-f)
[ ] SL directory (-s)
[ ] Record count to determine field descriptions (-n) 1
[ ] Unique prefix (-p)
[ ] Default number of occurs (-o) 1

Options: -d C:\AcuVet\FD

Close

When you invoke **xml2fd** from the Data view, the utility understands the context of the project and prefills the “-d” option entry field with the name of the directory in which the FD and SL are stored.

When **xml2fd** is invoked from the Tools menu, it does not have this context information, and therefore does not prefill the “-d” entry field.

2. If necessary, click the browse (“...”) button next to the “Output directory (-d)” field to navigate to the directory into which you want the FD and SL to be generated.

   Use the “-f” and “-s” options if you prefer to generate the FD and SL into separate directories.

   Note that selecting “-d” causes “-f” and “-s” to be disabled, because they are mutually exclusive options. Likewise, if “-f” or “-s” is selected, “-d” is disabled.

   In the absence of any instructions (“-d”, “-f”, or “-s”) about where to write the output, output is written to the current working directory, typically the root of the active project.

3. Set the “-n” option to specify the number of records to be read by the utility before it arrives at field descriptions in the FD. In very large XML files, it may not be desirable to read the entire file to derive the FD. On the other hand, it may be necessary to read more than one record to confirm a “best guess” for intended field descriptions in a record. The value of “-n” must be numeric and greater than “0”.

4. If you want to add a standard prefix to data items in the FD, use the “-p” option.

5. Use the “-o” option with a numeric value greater than “0” to instruct **xml2fd** to look for cases within the record where it is appropriate to assign an OCCURS clause. The value that you specify provides a default number to assign in the OCCURS clause. In complex cases where an FD has more than one OCCURS clause and where the number of OCCURS differs, users must manually change the generated representation of the FD in the graphical designer.

6. As you make changes in the XML2FD Utility window, the Options field displays the **xml2fd** command line that you are constructing. Verify this command line, then click **Start** to build the FD and SELECT.

   The Output window prints the output of **xml2fd**. More information about this utility can be found in section 8.2.2, “XML-to-FD Utility,” in *A Guide to Interoperating with ACUCOBOL-GT.*

20 The AcuBench Integrated Debugger

Key Topics

The Debugger Interface 20-2
Debug Mode Compile Options 20-7
Entering the Debugger 20-8
Debug Menu Commands 20-9

20.1 The Debugger Interface

The AcuBench integrated debugger is a source level debugger that runs inside the AcuBench workspace. The integrated debugger provides most of the functional capabilities of the ACUCOBOL-GT runtime debugger, as well as a few AcuBench-specific features. Most notably, the integrated debugger offers you the ability to modify your code during debug, without leaving the debugger interface.

Note: Users of Windows XP who have installed Service Pack 2 should be aware that the operating system's default firewall protection may block operation of the integrated debugger. When prompted to choose whether or not to keep blocking "AcuBench80", select **Unblock**.

This chapter describes in detail those elements of the debugger interface that are unique to AcuBench. It also notes equivalencies between AcuBench and runtime debugger functions. For more information about those equivalent functions, refer to the *ACUCOBOL-GT User's Guide*, section 3.1, "Runtime Debugger," which includes extended descriptions of debugger functions.

Note that if you prefer to use the ACUCOBOL-GT runtime debugger, instructions on how to access it from within AcuBench are included in **Chapter 9, section 9.8, "Debugging a Program."**.

Integrated debugger functions appear in a Code Editor window in the development area of the workspace. The debugger interface also uses the Output window, usually displayed at the bottom of the screen, to display various output messages generated by the debugger. These messages may include tracing information, information about breakpoints, error messages, and so on.

When using the integrated debugger, you may want to open the File tab of the Workspace window. The File tab provides a tree view representation of your project's Source, Screen, Report, Copy, Object, List, Resource, FD, and Remote files. (See **section 6.3.1, "Creating a Project,"** for more information about project files.)

You can toggle the Output and Workspace windows on and off from the View menu or the Standard toolbar, or close them by clicking the Close button on each window. You can also resize windows by dragging their borders with the mouse.

The AcuBench integrated debugger offers several features to assist you in debugging your code:

- The current line of execution is highlighted, and if the Line Number pane is visible, its line number is marked. You can toggle the highlighting and specify the highlight color in the Tools/Options/Environment/Debug dialog box, described in this manual in Chapter 4, **section 4.3.5, "Debug Options."**

- You can make changes to your code while debugging. The changes do not affect the current execution of the program, but you can test the changes simply by restarting the debugger, as described in this chapter in **section 20.4.2, "Starting, Stopping, and Navigating the Debugger."**

- Three specialized windows give you information about your code as you debug: Watch, Stack Info, and Memory. These windows can be displayed in the Output window or as floating windows, either one at a time or together, in any combination.

- A Quick Watch window lets you select any variable in the source code and display its current value.

- Using AcuConnect® with Micro Focus's Thin Client technology, you can debug a program that resides on a remote server using the same AcuBench debug commands that are used to debug a local program.

20.1.1 Debug Menu and Toolbar

AcuBench debug commands can be accessed from the Debug menu and the Debug toolbar. You can also create keyboard shortcuts for many debug commands. A complete list of user-definable shortcuts is found in **Chapter 23** of this manual. Descriptions of the debug commands appear later in this chapter.

As with other workbench toolbars, the Debug toolbar can be customized by selecting **Toolbars/Customize** from the View menu. To display or hide the Debug toolbar, select **Toolbars/Debug** from the View menu. You can also toggle the display by right-clicking in the toolbar area and choosing **Debug** or by selecting **Customize/Toolbars** from the Tools menu and then setting or clearing **Debug** in the Customize dialog box.

20.1.2 Debugger Output

Debugger messages appear in the Output window, which usually appears at the bottom of the AcuBench screen. If you have created an error file and are capturing trace information, you can also elect to display this information in the Output window.

Note that if you choose to send trace information to the Output window but no runtime error file exists, unusual screen behavior and additional undefined behaviors will result.

To send tracing information to the Output window:

1. Use the Project Settings dialog to establish an error file. On the Runtime tab, first select the **I/O options** catalog, then mark the **File to be opened for error message (-e)** check box. Don't forget to add an error file name in the entry field.
2. Close the Project Settings interface.
3. Open the Debug menu, expand the Trace Option sub-menu, and select **Trace to Debug Window**.

   Now, when you turn on the debugger's tracing options, all generated trace information is sent both to the error file and to the Output window.

20.1.3 Watch Window

The Watch window displays any program variables that you have selected. When you add a variable to the Watch window, the variable name appears in the Variable column, to the left of the variable's value. As the program executes, any changes to the values of the listed variables are displayed.

Note: Placing a watch on a variable does not cause the debugger to break when the value of the variable changes. If you want to monitor a variable and break when its value changes, use the Monitor tab of the Breakpoints window. See **section 20.4.4.5** for details.

After you start a program in the AcuBench integrated debugger, you can set a watch in either of two ways:

1. Use the **Watch Window** toolbar button to open the Watch window, or select **Debug Window/Watch** from the View menu. Double-click in a blank area of the Watch window to open a blank entry field, then enter the variable name.

   After you enter the name, the value displays automatically in the adjacent field.

2. Select the desired variable name in your code, then right-click and select **Add to Watch Window**. You can also invoke this command from the Variable List by selecting a variable name in the box, right-clicking, and choosing **Add to Watch Window**.

   Note that the Watch window must already be open in order for this command to have an effect.

Once you have added variables to the Watch window, you can choose whether their values are displayed in standard alphanumeric format or hexadecimal format. To toggle between formats, right-click anywhere in the Watch window and select or de-select the **Hexadecimal Display** check box in the pop-up menu.

The Watch window can also be used to ACCEPT a variable as you debug. With the debugger window active, simply type a new value for the variable in the Value column of the Watch window. Note that you are able to add variables or change values in the Watch window only after you have entered or re-entered the debugger. When the program is in an ACCEPT loop, the Watch window remains visible, but is not enabled for editing.

To remove a variable from the Watch window, highlight the variable and press the **Delete** key.

Quick Watch window

If the Watch window is not open, you can still retrieve the value of a variable by performing a quick watch. When you perform a quick watch, the selected variable and its value appear in a Quick Watch window. You can either close the Quick Watch window after viewing a variable's value, or click the **Add Watch** button to add the variable to the Watch window. If you add a watch on a variable when the Watch window is closed, AcuBench automatically opens the Watch window.

To perform a quick watch on a variable:

- Double-click the desired variable in the Code Editor to open a Quick Watch window.
- Highlight the desired variable in the Code Editor and click the **Quick Watch** toolbar button.
- Right-click the desired variable in the Code Editor and select **Quick Watch**.

You can view the values of up to seven previously selected variables by choosing from the drop-down box at the top of the Quick Watch window.

20.1.4 Stack Info Window

The Stack Info window is equivalent to the ACUCOBOL-GT debugger's View Perform Stack command. The window lists all of the nested paragraphs leading up to the current statement, from the beginning of the program or thread. The Stack Info window displays the source file name, line number, and paragraph.

By default, the Stack Info window displays the full path to a particular file. To toggle the display between the full path name and just the file name, right-click inside the Stack Info window and select **Full Path Filename**.

To position the cursor at a specific paragraph in your code, double-click the file path in the Stack Info window.

When you start a program in the debugger, you can view the Stack Info window either by choosing **Debug Window/Call Stack** from the View menu or with the **Call Stack** toolbar button. As your program executes, stack information (file path, line number, and paragraph name) appears in the window.

20.1.5 Memory Window

The Memory window offers the same features as the ACUCOBOL-GT runtime debugger's Memory Usage command, displaying the amount of program, file, window, overhead, and total memory used by your program, including standard and paged memory.

After starting your program in the debugger, open the Memory window by selecting **Debug Window/Memory** from the View menu or by using the **Memory Window** toolbar button. Memory status information is displayed automatically in the categories mentioned above.

20.2 Debug Mode Compile Options

In AcuBench, a mode is a set of compile options, runtime options, environment variables, and library options for a project. Debug Mode is one of two pre-defined modes in the workbench. In Debug Mode, the "Include all debugging information (-Ga)" compile option is automatically set, as well as the standard "Name of object file (-o)" and "Ignore CBLFLAGS environmental variable (-x)" options. For remote debugging, the "Name of remote object file (-o)" option is also set.

To select Debug Mode, use the **Project/Set Active Mode** command or select **Debug Mode** from the list box on the Project toolbar. You can also select **Debug Mode** in the Project Settings dialog box. More information about modes and how to add and delete them in AcuBench can be found in **Chapter 7, section 7.2, "Modes."**

AcuBench allows three levels of debugging: source, symbolic, and low. Source level debugging lets you view your source code while you are debugging. With symbolic debugging, your source code is hidden, but the

Output window displays any error messages and warnings. To access the low-level debugging function from within AcuBench, compile with the "-d" option. Refer to the *ACUCOBOL-GT User's Guide* for further information on these three levels of debugging.

Before you enter the AcuBench debugger, you should compile your program with at least one of the following compile options available from the Project Settings dialog box:

-Ga	Turns on all debugging options
-Gd	Includes the source code in the compiled object file
-Gl	Includes line number information
-Gs	Includes extra symbol information for symbolic debugging. This allows AcuBench to traverse the symbol table of the COBOL program and show group data items in a tree view control
-Gy	Includes minimal symbol information for basic symbolic debugging.

You can change debug compiler options in the Project Settings dialog box. Simply select the **Compiler** tab, choose **Debugger Options** from the catalog drop-down list box, clear the "-Ga" option, and select from among the other listed options.

20.3 Entering the Debugger

To enter the AcuBench debugger, select the program in either the File view or the Structure view of the Workspace window, then select **Go** from the Debug menu. You can also click the **Go** button on the Debug toolbar or use a shortcut key (**F8** by default). Note that the first time you select Go, the debugger opens the program source file and places the cursor at the first line of execution. When you select Go a second time, the program resumes execution from its current location. The program transfers control to the debugger when it reaches a breakpoint.

If your source file is already open in the Code Editor, you can launch the debugger by clicking on the editing window and selecting **Go** from the Debug menu.

Note that when your program is launched, a corresponding button is added to the Windows taskbar. When the program displays and accepts a screen, that screen does not automatically appear in the foreground. Click the button on the taskbar to shift focus from AcuBench to the running program.

Once a program has begun execution in the debugger, control is returned to the debugger in one of the following ways:

- When a breakpoint is reached. User-defined breakpoints are discussed in more detail later in this chapter.
- When a STOP statement executes that is not a STOPRUN. This behavior is similar to a breakpoint. Execution halts and the debugger cursor displays on a specific line. If you have not compiled with any debugging options, symbols and source are unavailable.
- When the debugger steps through the program and reaches the step count.
- When the Debug/Interrupt command is selected.
- When a monitored variable changes.

If the Output window is closed, it pops up over the lower portion of the screen when you enter the debugger. You can open it at other times by selecting **Output Window** from the View menu or by clicking the **Output Window** toolbar button. When you quit or exit the debugger, the Output window remains open until you close it by toggling the **Output Window** button or View menu command, or by clicking the window's **Close** button.

20.4 Debug Menu Commands

The AcuBench Debug drop-down menu includes all the commands needed for debug functions. AcuBench also allows you to define keyboard key-combination shortcuts for many debug commands. Refer to **Chapter 23, "Chapter 23: Keyboard Shortcut Reference,"** for more information. Debug commands are described in the following sections.

Note that while you are debugging, any function selected from the toolbar affects the active debugging session. For example, if you click **Go** while "Checkbox.cbl" is selected, and then select "Combo.cbl" and click **Go** a second time, the debugger behaves as though you have clicked **Go** a second time for "Checkbox.cbl". Compile and Execute functions are disabled during a debug session. Therefore, if you want to perform any build-related functions, you must first exit the debugger.

20.4.1 Tracing Functions

The Debug menu's **Trace Option** commands generate output messages that describe operations performed in a given domain during program execution. You can invoke the various trace functions only through the Debug menu's **Trace Option** commands. Each function is described briefly below. More information about trace functions can be found in section 3.1 of the *ACUCOBOL-GT User's Guide*.

Trace File toggles file tracing on and off. Output trace information includes all file-related operations performed at runtime. Trace level for this command can be set from 1 to 10 via the Trace Level dialog box described below.

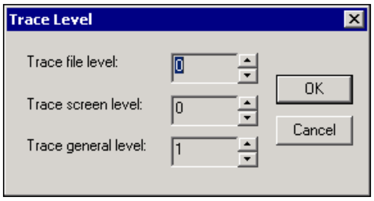
Trace Paragraphs toggles paragraph tracing on and off. Output trace information includes all paragraphs and sections entered at runtime.

Trace Screens toggles screen tracing on and off. Trace output includes information about DISPLAYs of Screen section items and about CREATEs, DISPLAYs, MODIFYs, and INQUIREs of ActiveX objects. Trace level for this command can be set from 1 to 10 via the Trace Level dialog box described below.

Trace Flush toggles trace flushing on and off if you are writing to an error file. With this command, the error file is flushed to disk after each WRITE operation.

Trace General toggles general tracing on and off. Trace output includes information on operations not covered by either the Trace File or Trace Screens command. Trace level for this command can be set from 1 to 10 via the Trace Level dialog box described below.

Trace Levels displays the Trace Level dialog box, in which you can set the level of detail in the output messages generated.



Trace file level: 0
Trace screen level: 0
Trace general level: 1

Trace levels can be set for the Trace File, Trace Screens, and Trace General commands. The higher the trace level, the greater the amount of debugging information sent to the error file. Therefore, an entry of “10” generates maximum output and an entry of “1” generates minimum output. An entry of “0” (the default) generates no output. The extra information is useful primarily to the ***extend*** Technical Support department.

To designate an error file to receive trace level output, select the **File to be opened for error message (-e)** option in the Project Settings/Runtime/I/O options dialog box. Indicate the name of the error file in the adjacent entry field. Note that if you set a trace level of 10, and see no output, this can mean that program execution included no output triggers.

Trace to Debug Window toggles the display of trace output messages in the Output window. The trace information displayed may be the result of any trace command except Trace Levels. You should establish an error file before you use this command. Refer to **section 20.1.2, “Debugger Output,”** for information about directing output to an error file.

20.4.2 Starting, Stopping, and Navigating the Debugger

The Debug menu contains a number of navigation commands used to start, stop, and control the execution of your program in the debugger. Some of these commands are assigned default keyboard shortcuts; others have not been assigned shortcuts. In either case, you can create your own customized keyboard shortcuts for the navigation commands as described in **section 4.3.6, “Keyboard Options.”**

Go, which is equivalent to the ACUCOBOL-GT debugger's Run/Continue command, starts the selected program in the debugger. If your files are out of date, you may be prompted to recompile them (depending on the build options that you have specified). If you select the Go command a second time, it resumes execution of your program from the cursor's current location. The program returns to the debugger when it reaches a breakpoint.

Restart ends the current debugging session and begins a new session with the same program. When you select Restart, the debug cursor returns to the first statement in the code. Restart is useful when you want to repeat the session or when you have altered lines of code and want to rebuild and begin a session from the first statement.

To have the debugger automatically rebuild your program when you restart, select **Follow the options specified in the 'Build' page** in the Environment/Debug section of the Tools/Options window. To customize the Build rules for your project, go to the Environment/Build section of the Tools/Options window.

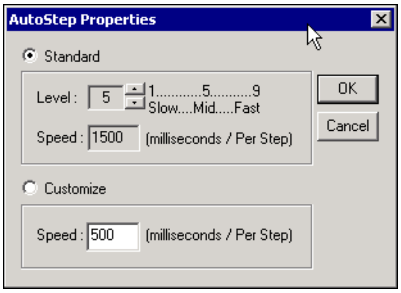
Exit Debugger halts the current debugging session but leaves the debugger open. It is equivalent to the ACUCOBOL-GT File/Exit command.

Quit Debugging halts the debugging session and exits the debugger. It is equivalent to the ACUCOBOL-GT File/Quit command.

Interrupt stops execution of the application and returns control to the debugger, just like the Ctrl+Break shortcut. It creates a pause similar to that caused by a breakpoint, with one caveat: if the program is in a PERFORM or ACCEPT loop, the Interrupt command is not received until an exception occurs or the action completes.

After receiving the Interrupt command, the debugger cursor displays on the current line of execution. Note that you should use Go, rather than Restart, to continue debugging after using the Interrupt command.

Auto Step lets you execute "step" commands repeatedly until the program's end. You can change the speed at which the command executes, from "1" (slowest, approximately 3 seconds per step) to "9" (fastest, several steps per second). When you select a speed level in the "Standard" area of the dialog, the actual speed is displayed in milliseconds per step. You can also use the "Customize" area to set a custom speed, specified in milliseconds per step.


AutoStep Properties

(● Standard) Level : 5 1...........5..........9 Slow....Mid.....Fast

Speed : 1500 (milliseconds / Per Step)

Customize

Speed : 500 (milliseconds / Per Step)

Step Into allows you to step to the next statement in your program. You can use this command to step into a new thread when it is created. If you want to continue to track the original thread, use the debugger's Step Over command.

Step Over allows you to step over the next statement in your program. Use this command if you want to track a single thread rather than stepping into a new thread each time one is created.

Step Out Paragraph runs your program until it returns to the point at which the command was invoked. It is equivalent to the ACUCOBOL-GT debugger's Run/Go until Paragraph Returns command.

Step Out Program runs your program until it exits to its calling program. It is equivalent to the ACUCOBOL-GT debugger's Run/Go until Program Exits command.

Skip to Line moves the current program location to the line containing the cursor. Further execution of your program proceeds from this line. The current program location does not change unless the cursor line contains a verb. This command is equivalent to the ACUCOBOL-GT debugger's Run/Skip to Cursor Line command.

Use this command with care, because the skipped lines are not executed. You may skip important sections of code and experience unexpected results.

Run to Cursor sets a temporary breakpoint at the current cursor line (or the closest previous line with a verb) and continues execution of your program. It is equivalent to the ACUCOBOL-GT debugger's Run/Go to Cursor Line command.

20.4.3 Debugger Scripts

A debugger script file contains keyboard input and menu selection information to automate the process of debugging a program.

To record a script, select **Record Script** from the Debug menu. This command is equivalent to the ACUCOBOL-GT debugger's File/Record Script command, and turns on a recorder that saves all of your keyboard input and menu selections to the file that you specify.

When the recorder is running, the Record Script menu option is replaced by a Stop Recorder option. Use this to end your recording. If you do not end your recording, nothing is saved in the file.

While the recorder is active, you cannot use the mouse for anything except selecting menu items.

The recorder can save up to 4096 characters of information. Normal keystrokes use one character. Special keys such as function keys and menu selections can use up to four characters.

To run a debugger script file, select **Run Script** from the Debug menu. This causes all input (debugger and program) to be read from the script. Control returns to the keyboard when the script is finished. This command is equivalent to the ACUCOBOL-GT runtime debugger's File/Run Script command.

20.4.4 Breakpoints

A breakpoint is a location in your program's code that you designate. When a breakpoint location is reached during program execution, control is returned to the debugger before the code at the breakpoint is executed.

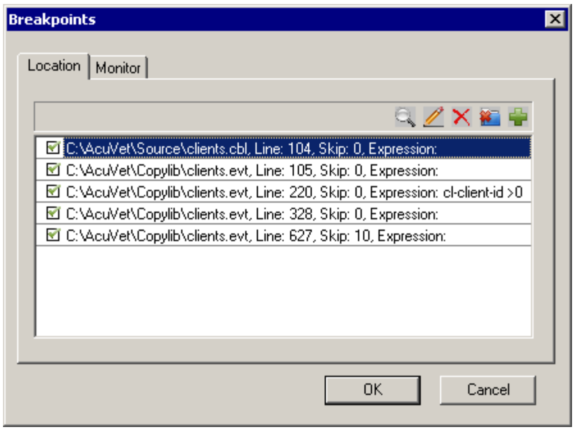
Enabled breakpoints appear as red thumbtack icons in the Line Number pane of the debugger's Code Editor window. Disabled breakpoints appear as yellow thumbtack icons.

Bookmarks that you set in your source code when you are working in the Code Editor automatically become breakpoints when you enter the debugger. These locations are marked with a standard bookmark icon, rather than the breakpoint thumbtack. Note that bookmarks are not included in the list of breakpoints displayed in the Debug/Breakpoints dialog box.

Breakpoints are most easily set and managed using the Debug toolbar buttons.

- To set or remove a breakpoint, use the **Toggle Breakpoint** push button.
- To temporarily disable a breakpoint, use the **Disable Breakpoint** push button.
- To remove all breakpoints, use the **Clear All Breakpoints** push button.

You can also set, modify, and delete breakpoints using the Breakpoints dialog box. To do this, select **Breakpoints** from the Debug menu.



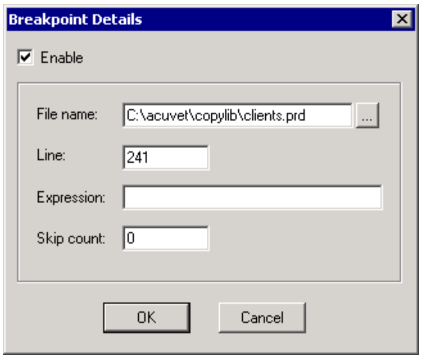
☑ C:\AcuVet\Source\clients.cbl, Line: 104, Skip: 0, Expression:
☑ C:\AcuVet\Copylib\clients.evt, Line: 105, Skip: 0, Expression:
☑ C:\AcuVet\Copylib\clients.evt, Line: 220, Skip: 0, Expression: cl-client-id >0
☑ C:\AcuVet\Copylib\clients.evt, Line: 328, Skip: 0, Expression:
☑ C:\AcuVet\Copylib\clients.evt, Line: 627, Skip: 10, Expression:

The Breakpoints dialog displays a list of all the breakpoints set in your code, including the file name, line number, skip count, and expression associated with the breakpoint, as well as whether the breakpoint is currently enabled. With the buttons in the upper right corner of the tab, you can view the code associated with a breakpoint, modify or delete a breakpoint, delete all breakpoints, or add a new breakpoint. Selecting the New or Modify button opens the Breakpoint Details dialog. The Breakpoint Details dialog allows you to enter or modify the file name, line, skip count, and expression of the breakpoint. You can also enable or disable a breakpoint and set monitors on variables using this interface.

All of these functions are described in the following sections.

20.4.4.1 Setting a breakpoint using the Breakpoints dialog box

1. Select **Breakpoints** from the Debug menu to open the Breakpoints dialog box.
2. On the Location tab, click **New** (the button in the top, right corner of the tab) or double-click in the window.



Enable: ✓
File name: C:\acuvet\copylib\clients.prd
Line: 241
Expression:
Skip count: 0

The Breakpoint Details dialog box appears. By default, the “Enable” check box is selected.

3. Fill in the file name, line number, expression, and skip count for your breakpoint. Use the **Browse** push button if necessary to locate your file name.
4. Click **OK** to return to the Breakpoints dialog. The location of your new breakpoint appears in the dialog box.
5. Click **OK**.

20.4.4.2 Modifying a breakpoint using the Breakpoints dialog box

1. Select **Breakpoints** from the Debug menu to open the Breakpoints dialog box.

   The Location tab displays a list of the current breakpoint locations, along with each breakpoint's file name, line number, skip count, and expression.

2. Click **Modify** (the second of the five buttons in the top, right portion of the tab).

   The Breakpoint Details dialog box appears. Here, you can enable or disable a breakpoint; change the file name; or alter the line number, expression, or skip count.

3. Click **OK** to return to the Breakpoints dialog box. The modifications to your breakpoints are now reflected in the Breakpoints list.

4. Click **OK**.

20.4.4.3 Deleting breakpoints using the Breakpoints dialog box

1. Select **Breakpoints** from the Debug menu to open the Breakpoints dialog box.

2. On the Location tab of the dialog box, select the breakpoint you want to delete.

3. Click **Delete** (the third of the five buttons in the top, right portion of the tab). To remove all your breakpoints, click **Delete All** (next to the New button).

4. Click **OK**.

20.4.4.4 Viewing a breakpoint with the Breakpoints dialog box

1. Select **Breakpoints** from the Debug menu to open the Breakpoints dialog box.

2. On the Location tab of the dialog box, select the breakpoint location you want to view.

3. Click **View Source** (the first of the five buttons in the top, right portion of the tab).

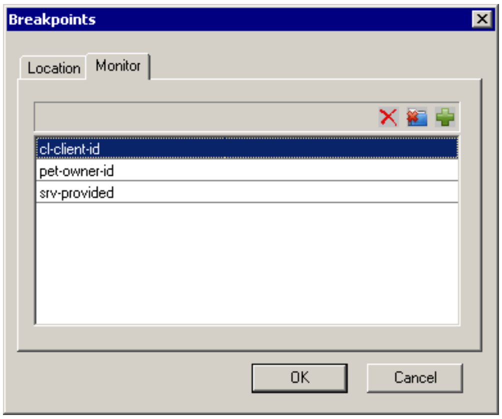
   This command positions the cursor at the indicated breakpoint in your source code.

20.4.4.5 Monitoring variables using the Breakpoints dialog box

The Monitor tab of the Breakpoints dialog box lets you monitor value changes in the variables in your program. The Monitor tab lists all the variables that you want to monitor, and allows you to add variables to the list or delete variables from the list.

To monitor a variable while debugging:

1. Select **Breakpoints** from the Debug menu to open the Breakpoints dialog box, then select the Monitor tab.



cl-client-id
pet-owner-id
srv-provided

2. To add a new variable to monitor, click **New** or double-click in the window.

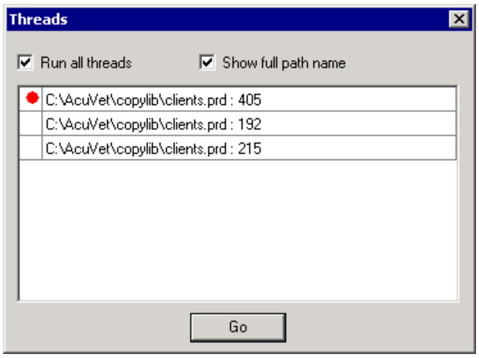
3. Enter the name of the variable to monitor.

4. Click **OK**.

You can use the **Delete** and **Delete All** push buttons to remove monitors on variables.

20.4.5 Debugging Threaded Applications

The Debug menu's **Threads** option opens the Threads dialog box, which displays a list of the threads in the program being debugged. This list includes the file name and the location of each thread.



☑ Run all threads ☑ Show full path name

```
● C:\AcuVet\copylib\clients.prd : 405
  C:\AcuVet\copylib\clients.prd : 192
  C:\AcuVet\copylib\clients.prd : 215
```

By default, the "Run all threads" check box is enabled, which means the debugger runs all threads simultaneously.

- Use the **Step Into** command to step into a new thread when it is created.
- Use **Step Over** when you want to trace a single thread in your program.

  Note that breakpoints in other threads remain active and can transfer control to the debugger. When a thread other than the current thread returns control to the debugger, that thread becomes the current thread.

To execute only one thread at a time, disable the **Run all threads** check box and press **Go**. In this case, to switch between threads, select the thread you want from the Threads dialog box and click **Go**.

If you enable the "Show full path name" check box in the Threads dialog box, the list of threads includes the file's full path. Disable this check box if you want the dialog box to list only the file name.

While the Threads dialog provides a useful tool for controlling how the debugger navigates between threads, the act of single-stepping through a program, by its nature, interferes with the natural threading behavior of the runtime. As a result, if you are trying to debug a threading issue, you may receive a more accurate picture of your program's behavior by enabling paragraph tracing and running the program with an error file than by stepping through the program line-by-line.

20.4.6 Quick Watch

If you select a variable in the Code Editor and then use the Debug/Quick Watch command, a Quick Watch window appears, displaying the variable's current value. For more information about the Quick Watch window, refer to **section 20.1.3, "Watch Window."**

21 Looking for Something?: Search and Replace

Key Topics

Introduction 21-2
Find 21-2
Find in Files 21-4
Find in Objects 21-7
Replace 21-8
Replace in Files 21-9

21.1 Introduction

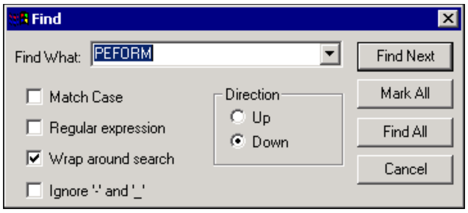
When you are working in AcuBench, a variety of search and replace options are available to allow you to locate a text string within the current document, within the AcuBench graphical designers associated with a project, or within the files in a specified directory structure.

21.2 Find

To search for a string in the document currently open in the Code Editor or Event Editor, do one of the following:

- Open the Edit menu and select **Find**.
- Right-click in the editor window and select **Find**.
- Use the default **Ctrl + F** keyboard shortcut.

This opens the Find dialog, in which you can enter a search string and set a variety of options for the search.



Find What: PEFORM
[ ] Match Case
[ ] Regular expression
[✓] Wrap around search
[ ] Ignore '-' and '_'
Direction: Up [✓ Down]

The **Find What** combo box displays the string to be used in the search. If text is selected in the editor when you open the Find dialog, that text will appear selected in the combo box. If you are performing multiple searches, you can expand the combo box to see and select from a list of up to 15 previous search terms. You can also type the search term that you would like to use in this field.

If you want to find only those strings that exactly match the case of the search string that you enter, mark the **Match Case** check box. If this box is not marked (the default), the search is not case-sensitive.

If you want to search for a regular expression, rather than a text string, mark the **Regular Expression** check box. A regular expression is a formula for matching strings that have a certain pattern. Regular expressions are discussed in more detail in the next section, "**Find in Files**."

If you are not using regular expressions, and you do not want the search function to consider hyphens and underscores, mark the **Ignore "-" and "_"** check box.

If you want AcuBench to search the entire document, regardless of where the cursor is positioned when you start the search, mark the **Wrap Around Search** check box. This means that in a "down" search, when the search reaches the end of the active document, it continues searching from the top. Likewise, in an "up" search, when the search reaches the beginning of the active document, it continues searching from the end.

To determine the direction for the search, select either the **Up** radio button, to search from the cursor position to the top of the file, or the **Down** radio button, to search from the cursor position to the end of the file.

21.2.1 Locate a single instance of the search string

To find a single instance of the search string, click **Find Next** to start the search function. The first instance of the string that AcuBench locates is highlighted. To continue to the next instance of the string, click **Find Next** again.

21.2.2 Mark all instances of the search string

If you expect that multiple instances of a string will be found, and you want an easy way to refer to each instance, click **Mark All** to perform the search. AcuBench places a bookmark at each line on which the string is found. Navigate between bookmarks with the Previous Bookmark and Next Bookmark Edit commands, toolbar buttons, or keyboard shortcuts.

21.2.3 List all occurrences of the search string

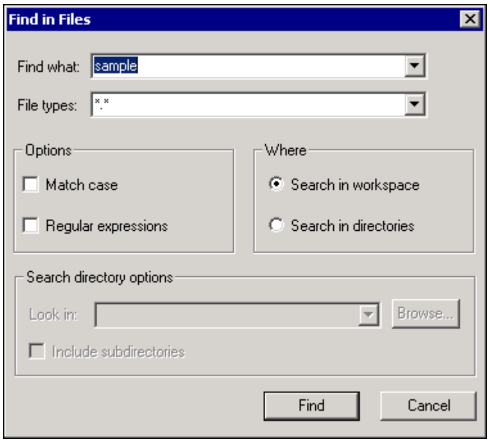
Alternatively, to see a list of all instances of a search string in the Output window, click **Find All** to begin your search. A list is generated, showing the file path, line number, and text of each line in the active document that contains the string. Double-click on item in the list to jump to that point in the active document.

21.3 Find in Files

To locate all instances of a search string within a specified directory or set of directories, do one of the following:

- Open the Edit menu and select **Find in Files**.
- Click the **Find in Files** button on the Editor toolbar.

When you choose this command, the Find in Files dialog box appears.


Find in Files

Find what: sample

File types: *.*

Options
- [ ] Match case
- [ ] Regular expressions

Where
- (●) Search in workspace
- ( ) Search in directories

Search directory options
- Look in:
- [ ] Include subdirectories

Like the basic Find dialog, the Find in Files dialog box starts with a **Find what** combo box that displays any currently selected text and provides a list of previous search commands.

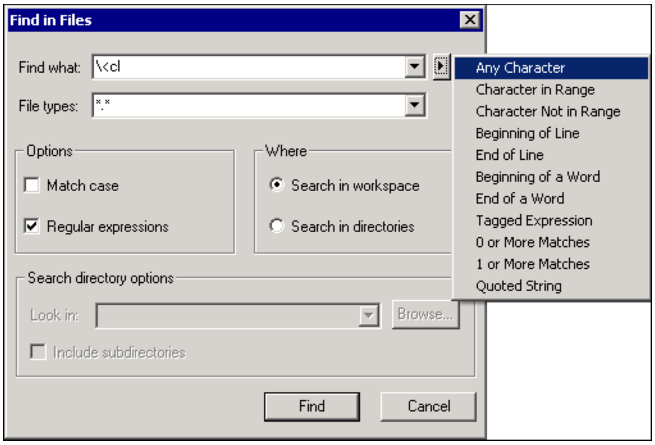
In addition, because this search function searches specified directories for files containing a text string, rather than limit itself to the active document, you can limit the search to specific types of files. Use the **File types** field to enter one or more file extensions. Only files with the specified extension(s) will be examined during the search. The default is to search all file types ("*.*").

As in the basic Find dialog, you can choose whether or not the search should be case-sensitive (the **Match case**) option, and whether or not regular expressions are used.

21.3.1 Special Operators and Regular Expressions

A regular expression is a formula for matching strings that have a certain pattern. It contains both normal alphabetic and numeric characters and *metacharacters*. Metacharacters are used to create the template pattern used in the search.

When you mark the **Regular expressions** check box, a button appears on the Find in Files interface, to the right of the "Find what" field. When you click this button, a pop-up list displays descriptions of various supported metacharacters.


Find in Files

Find what: \<cl

File types: *.*

Options: Match case [ ], Regular expressions [✓]

Where: [● Search in workspace] ○ Search in directories

Search directory options

Look in:

Include subdirectories [ ]

Select a description from the list and the metacharacter is added to the "Find what" field. If you are familiar with Windows regular expression rules, you can also type in your own patterns.

The metacharacter options available from the pop-up list include:

Any Character .	When the search function looks for a match for the specified string, any character is acceptable in this character position.
Character in Range []	Any of the characters between the brackets counts as a match for the search function.
Character Not in Range [^]	Any character *other* than those listed between the brackets is an acceptable match.
Beginning of Line ^	Find the preceding search string only when it occurs at the beginning of a line.
End of Line $	Find the preceding search string only when it occurs at the end of a line.
Beginning of Word \<	Find the following search string only when it occurs at the beginning of a word.
End of Word \>	Find the preceding search string only when it occurs at the end of a word.
Tagged Expression \(\)	Treat the expression between \(and \) as a group.
0 or More Matches *	Find zero or more occurrences of the preceding character.
1 or More Matches +	Find one or more occurrences of the preceding character.
Quoted String \	This is an escape character, used before a symbol that may be used as a metacharacter to indicate that it is being used as a literal. For example, if you were searching for any library routine, you could use "\$" to indicate the dollar sign character, rather than the end of line metacharacter.

Some sample expressions using regular expressions include:

^…$	Any line having three characters
[0-9]	Any digit
[1-2].$	Strings where the second-to-last digit is “1” or “2”
[a-z]	Any lowercase alphabetic character
^[A-C]	Strings where the first character is an uppercase “A”, “B”, or “C”
[Ww]indow	The strings “Window” or “window”

One useful, supported metacharacter not included in the regular expressions pop-up list is the pipe symbol (“|”). This is used between two separate search strings or search patterns to indicate that either of the two expressions is acceptable. The expression “if|else”, for example, will find any line that contains either the word “if” or the word “else”.

21.3.2 Performing the Search

Two radio buttons let you choose whether to search through the files associated with the current workspace or files located in a specified directory path. When you select **Search in directories**, the Search directory options frame is enabled. You can enter a directory path, or click the **Browse** button to navigate to a directory. Mark the **Include subdirectories** box to include any subdirectories within the specified path, or uncheck the box to search only the specified directory.

When you have specified the location to be searched, click **Find**. AcuBench searches the specified files and folders, displaying all matches in the Output window. To see one of the specified instances of the string, double-click the corresponding line in the Output window. AcuBench opens the selected file in the Code Editor and places the cursor at the specified line.

21.4 Find in Objects

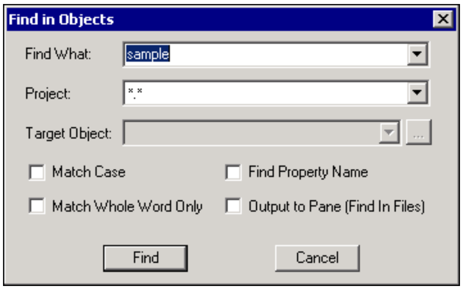
When you are working with programs that appear in the Structure view, you may find it useful to search for a given string within the various AcuBench designers used by the program. For example, you may want to search for a variable defined in the Screen Designer and referenced in the Event Editor. In this instance, Find in Files would locate the string in the read-only COPY file associated with the designer, rather than the actual designer. If you want to make changes to the string, this presents some difficulties.

To help you make efficient use of the AcuBench graphical designers, the Find in Objects function helps you to locate all instances of a given string stored in the Property window and the graphical design interfaces.

To search the graphical designers for all instances of a string, do one of the following:

- Open the Edit menu and select **Find in Objects**.
- Right-click in any of the graphical designers and select **Find in Objects**.

When you issue this command, the Find in Objects dialog opens.



Find What: sample
Project: *.*

As with the other Find dialogs, the **Find What** combo box is used to display the selected search string. The most recent search term, if any, is displayed in the entry portion of the combo box. You can expand the box to see a list of up to 15 previous search terms.

When you open the Find in Objects dialog at the project level, rather than from within an individual designer, you will see a **Project** combo box under the Find What field. If your workspace contains only one project, the name of that project will appear here. If your workspace contains multiple projects, by default, "*.*" appears here, indicating that all projects in the workspace should be searched. To search a single project in a workspace containing multiple projects, expand the combo box and select the project from the list.

When a single project name appears in the Project combo box, the **Target Object** list is enabled. This allows you to choose whether to search the objects associated with all PSF and DLT files in the project, or to search only the objects associated with certain PSF and DLT files.

Regardless of the context in which you launch the Find in Objects dialog, you can choose whether or not the search is case-sensitive with the **Match Case** option. You can also use the **Match Whole Word Only** option to determine whether the specified string should be considered only as an entire word, or as either an entire word or a portion of a word. If you want the search operation to include screen and report properties, rather than only property values, mark the **Find Property Name** check box.

Finally, if for any reason you want the results of your search output to the "Find in Files" tab of the Output window, rather than the default "Find in Objects" tab, mark the **Output to pane (Find in Files)** option.

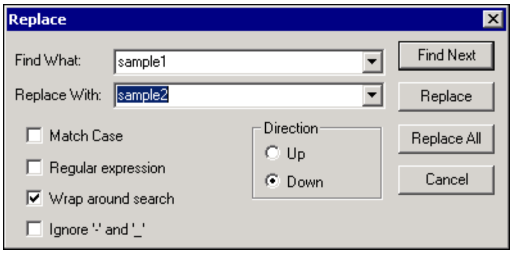
When you have finished filling in the dialog options, click **Find** to begin your search. Any search items found are listed in the Output window. Double-click an item in the list to open the corresponding graphical designer with the found item selected.

21.5 Replace

To replace instances of a string in a document currently open in the Code Editor or Event Editor, do one of the following:

- Open the Edit menu and select **Replace**.
- Click the **Replace** button on the Editor toolbar.
- Use the default **Ctrl + H** keyboard shortcut.

When you issue the Replace command, the Replace dialog box is displayed.



As with the Find dialogs, the Replace dialog includes a **Find What** combo box, used to display the current search term. Expand the combo box to see a list of up to 15 previous search terms.

Enter the replacement string in the **Replace With** combo box. A list of up to 15 previous replacement strings can be seen by expanding the combo box.

See **section 21.2** for more information about the check-box and radio button options at the bottom of the Replace interface.

When you have finished making your selections, click **Find Next** to locate the next instance of the search string. Select **Replace** to replace only that instance of the string or **Replace All** to replace all instances of the string. To skip an instance of the string and continue, select **Find Next**. To stop searching, click **Cancel**.

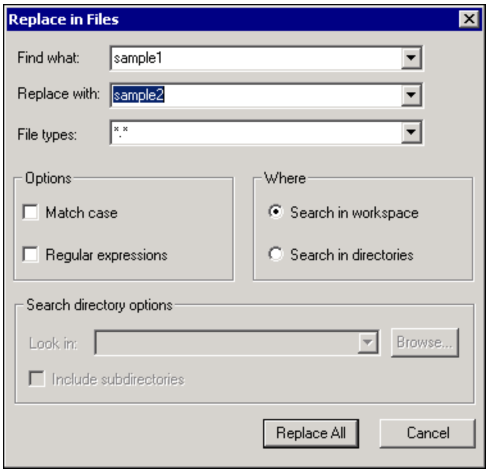
21.6 Replace in Files

Just as Find in Files allows you to search for instances of a given string within a specified workspace or directory structure, Replace in Files provides a way to replace all instances of a string within a workspace or directory structure.

To replace a search string in all files in the specified location do one of the following:

- Open the Edit menu and select **Replace in Files**.
- Click the **Replace in Files** button on the Editor toolbar.

When you select this option, the Replace in Files dialog appears.



Find what: sample1
Replace with: sample2
File types: *.*
Options: Match case [ ], Regular expressions [ ]
Where: [✓ Search in workspace] Search in directories
Include subdirectories [ ]

Most options on this screen correspond to options on the Find in Files interface, discussed at length in **section 21.3**. There is, however, one important thing to note about this interface: there is no option to replace a single item. When you choose **Replace All**, AcuBench replaces all instances of the text string without prompting for user confirmation.

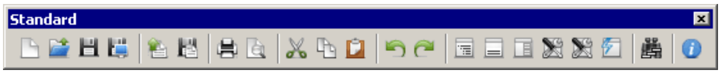
22 Toolbar Reference

Key Topics

The Standard Toolbar 22-2
The Project Toolbar 22-4
The Editor Toolbar 22-6
The Debug Toolbar 22-6
The Align Toolbar 22-8
The Launch Toolbar 22-8

22.1 The Standard Toolbar

The default Standard toolbar is shown below:



The Standard toolbar commands (looking at the toolbar illustration from left to right) are described in the following table:

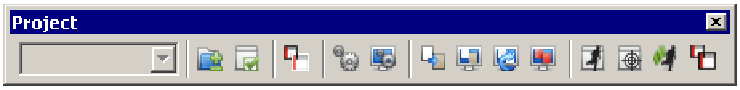
Description	Menu/Command Equivalent
New. Generates the New dialog box, from which you can open a new project, program, screen file, and source file. It also provides access to the File Designer interface.	**File/New**
Open. Lets you open an existing file. When you choose this command, the Open dialog box appears.	**File/Open**
Save. Saves the active file using its current file name. If the file has not yet been saved, the Save As dialog box appears.	**File/Save**
Save All. Lets you save all open files. You can save the files in new folders or in the same folders using a new name.	**File/Save All**
Open Project. Lets you open an existing project file (".pjf"). When you choose this command, the Open Project dialog box appears.	**File/Open Project**
Save Project. Lets you save a project file (".pjf"). If the project has not yet been saved, the Save Project dialog box appears.	**File/Save Project**
Print. Lets you send the active file to a printer. When you choose this command, the Print dialog box appears.	**File/Print**

Description	Menu/Command Equivalent
Print Preview. Lets you see how the active file looks before you print it. When you choose this command, the Print Preview screen appears.	**File/Print Preview**
Cut. Removes the selected text or screen element and places it on the clipboard.	**Edit/Cut**
Copy. Places a copy of the selected text or screen element on the clipboard.	**Edit/Copy**
Paste. Places a selection that has been cut or copied to the clipboard into the active file. When you choose this command, selected text is pasted to the cursor location, and a selected control is pasted to the same location that it occupies in the old screen.	**Edit/Paste**
Undo. Reverses recent editing actions. The workbench has multiple undo support, so previous commands can be undone.	**Edit/Undo**
Redo. Reverses recent undo operations if you have performed no other actions since the undo. The workbench has multiple redo support, so previous commands can be redone.	**Edit/Redo**
Workspace. Toggles the view of the workspace in the workbench window.	**View/Workspace**
Output Window. Toggles the view of the Output window in the workbench window.	**View/Output Window**
Property Window. Toggles the view of the Property window in the workbench window.	**View/Property Window**
Component Toolbox. Toggles the view of the Screen Component Toolbox in the workbench window.	**View/Screen Component Toolbox**
Report Component Toolbox. Toggles the view of the Report Component Toolbox in the workbench window.	**View/Report Component Toolbox**

Description	Menu/Command Equivalent
Toggle Drag-and-Drop. Toggles the view of the Drag and Drop interface.	**View/Drag-and-Drop**
Find in Objects. Lets you locate all instances of a given string stored in the Property window and the graphical design interfaces, such as the Working Storage Editor or the File Designer.	**Edit/Find in Objects**
About. Displays application identification and copyright information.	**Help/About**

22.2 The Project Toolbar

The default Project toolbar is shown below:



The Project toolbar commands (looking at the toolbar illustration from left to right) are described in the following table:

Description	Menu/Command Equivalent
Set Active Mode. Allows you to select a mode (set of compiler, runtime, and environment settings) from the drop-down list.	**Project/Set Active Mode**
Add/Remove Files. Allows you to add files to or remove files from a project.	**Project/Add/Remove Files**
Settings. Opens the Project Settings dialog. If a source file has focus, the dialog is adjusted to support file-level settings (compile options only).	**Project/Settings**

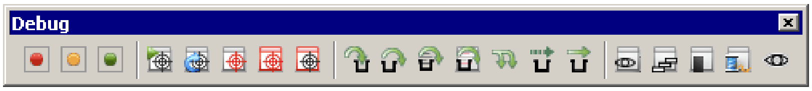
Description	**Menu/Command Equivalent**
Create Alias. Opens the Create Alias dialog, in which you can define the alias that contains the information needed for a server-based runtime to run a remote program in ACUCOBOL-GT® Thin Client operations.	**Project/Create Alias**
Generate. Generates code for the structural component of the program that has focus (screen, data layout, Working Storage Editor, etc.).	**Build/Generate**
Regenerate Workspace. Regenerates all code-generating elements in a workspace.	**Build/Regenerate Workspace**
Compile. Compiles the selected file.	**Build/Compile**
Build. Initiates a conditional compilation of the workspace. Only those files that have changed since the last build, and their dependents, are recompiled.	**Build/Build Workspace**
Rebuild. Initiates a full recompilation of the workspace. All files are compiled.	**Build/Rebuild Workspace**
Stop Build. Terminates the active Build or Rebuild process.	**Build/Stop Build**
Execute. Executes the selected program.	**Build/Execute**
Debug (Runtime). Starts the associated COBOL program in the ACUCOBOL-GT runtime debugger.	**Build/Debug (Runtime)**
Allow Parameters. Prompts for parameters before executing the selected program.	**Build/Allow Parameters**
Use Thin Client. Signals the workbench that you want to use ACUCOBOL-GT Thin Client related commands and project setting options.	**Build/Use Thin Client**

22.3 The Editor Toolbar

The Editor toolbar is used to invoke search and replace functions, as well as various Code Editor tools and commands. More information about using this toolbar can be found in **Chapter 12, section 12.4, "Basic Editor Functions."**

22.4 The Debug Toolbar

The default Debug toolbar is shown below:



The Debug toolbar commands (looking at the toolbar illustration from left to right) are described in the following table:

Description	**Menu/Command Equivalent**
Toggle Breakpoint. At the cursor position, sets or removes a breakpoint.	**Debug/Breakpoints**
Disable Breakpoint. Disables the selected breakpoint.	**Debug/Breakpoints**
Clear All Breakpoints. Removes all breakpoints from the program.	**Debug/Breakpoints**
Go. Starts the selected program in the AcuBench integrated debugger.	**Debug/Go**
Restart. Ends the current debugging session and begins a new session.	**Debug/Restart**
Exit Debugger. Terminates the current debugger session, but leaves the debugger open.	**Debug/Exit Debugger**
Quit Debugging. Terminates the debugger session and exits the debugger.	**Debug/Quit Debugging**

Description	**Menu/Command Equivalent**
Interrupt. Stops execution of the program, returning control to the debugger.	**Debug/Interrupt**
Step Into. Steps to the next statement.	**Debug/Step Into**
Step Over. Steps over the next statement.	**Debug/Step Over**
Step out Paragraph. Runs the program until the current paragraph returns to the point from which it was performed.	**Debug/Step Out Paragraph**
Step out Program. Runs the program until it exits to its calling program.	**Debug/Step Out Program**
Auto Step. Lets you execute “step” commands repeatedly until the program’s end.	**Debug/Auto Step**
Skip to Line. Moves the current program location to the line containing the cursor.	**Debug/Skip To Line**
Run to Cursor. Sets a temporary breakpoint at the current line (or closest line with a verb) and continues execution of the program.	**Debug/Run To Cursor**
Watch Window. Causes the Watch window to be displayed or hidden.	**View/Debug Window/Watch**
Call Stack Window. Causes the Stack Info window to be displayed or hidden.	**View/Debug Window/Call Stack**
Memory Window. Causes the Memory window to be displayed or hidden.	**View/Debug Window/Memory**
Threads Window. Generates the Threads dialog box.	**View/Debug Window/Threads**
Quick Watch. Opens the Quick Watch window, which displays the value of a selected variable.	**Debug/Quick Watch**

22.5 The Align Toolbar

The Align toolbar is discussed in detail in **Chapter 14**.

22.6 The Launch Toolbar

The Launch toolbar is a configurable toolbar for holding program icons linked to frequently used applications (such as a browser, spreadsheet, or word processing application). When you first install AcuBench, the Launch toolbar is empty. For step-by-step instructions on how to add icons to the Launch toolbar, see **Chapter 4, section 4.9, "The Customize Dialog."**.

23 Keyboard Shortcut Reference

Key Topics

Introduction 23-2
Main: Default Keyboard Shortcuts 23-3
Code Editor: Default Keyboard Shortcuts 23-24
Screen Designer: Default Keyboard Shortcuts 23-30

23.1 Introduction

AcuBench allows you to define keyboard key-combination shortcuts for most workbench commands. A list of all AcuBench keyboard shortcuts, organized by category, is contained in a drop-down list located in the Tools/Options/Environment/Keyboard dialog box. The Main, Code Editor, and Screen Designer categories indicate the state in which a particular command functions. The Main category, which lists the menu bar, toolbar, and window/right-click commands, is a superset of the Code Editor and Screen Designer categories. The Code Editor and Screen Designer categories each list shortcuts associated with their particular state. For example, you access the list of special keystrokes that perform commands particular to the Code Editor state by selecting the Code Editor category.

Using the three different categories, you can assign three different keyboard shortcuts to a common command (one for each category). When you assign different shortcuts to the same command name, AcuBench applies these shortcuts in context. Let's say you assign **Ctrl+P** to the Main category's FilePrint command and assign different shortcuts for FilePrint in the Code Editor and Screen Designer categories, **Ctrl+F1** and **Ctrl+F2**, respectively. If you begin working in the Screen Designer, **Ctrl+F2** is the shortcut key for FilePrint. If you change states and begin working in the Code Editor, then the shortcut key associated with FilePrint changes to **Ctrl+F1**.

Following is a brief description of the Main, Code Editor, and Screen Designer shortcuts and their default values. You can assign key combinations for commands without shortcuts or replace default shortcut key designations with your own definitions, using the procedures described in **section 4.3.6**.

Three common keyboard shortcuts are defined by the Windows operating system and cannot be changed. They are:

Close Document Window	**Ctrl+F4**
Next Document Window	**Ctrl+F6**
Previous Document Window	**Ctrl+Shift+F6**

23.2 Main: Default Keyboard Shortcuts

Command	**Shortcut Key**	**Description**
AboutAcuBench	undefined	Display version, copyright, license, and Micro Focus contact information
ActiveXControl	undefined	Display the ActiveX Control Components dialog box.
ActiveXProperty	undefined	Display the ActiveX control's properties
AddFD	undefined	Add an existing data layout (".dlt") file to the current project.
AddIOParagraph	undefined	In the IO Handling tab of the File Designer, add a new user-defined item to the paragraph list.
AddItem	undefined	Add a data item to the record definition in the File Designer.
AddItemBefore	undefined	Add a data item before the selected item in the File Designer's record definition.
AddKey	undefined	Add a key for an indexed file in the File Designer.
AddProgram	undefined	Add an existing program structure (".psf") file to the current project.
AddScreen	undefined	Add an existing screen (".smf", ".stf") file to the current program.

Command	**Shortcut Key**	**Description**
AddSubItem	undefined	Add a subitem to a group data item in the File Designer’s record description.
AdjacentHorizontal	undefined	Align controls to be adjacent horizontally.
AdjacentVertical	undefined	Align controls to be adjacent vertically.
AlignToGrid	undefined	Align control(s) to the grid.
ApplicationExit	**Ctrl+E**	Quit the application, and prompt to save the documents.
AssociateFDSL	undefined	Associate the named FD/SL files within the project.
BookmarkClearAll	undefined	Clear all bookmarks.
BookmarkNext	undefined	Move the cursor to the line containing the next bookmark
BookmarkPrev	undefined	Move the cursor to the line containing the previous bookmark.
BookmarkToggle	undefined	Toggle the display of a bookmark on the current line.
Build	**F7**	Build the project.
BuildCompile	**Ctrl+F7**	Compile the source file.
BuildDebug	undefined	Execute the selected program using the runtime debugger.
BuildExecute	**Ctrl+F5**	Execute the selected program.
BuildAllowParameters	undefined	Execute the selected program, allowing users to define parameters.

Command	**Shortcut Key**	**Description**
BuildProject	undefined	Build the project.
BuildStop	**Ctrl+Break**	Stop the build.
CenterHorizontal	undefined	Center the controls horizontally within the screen form.
CenterVertical	undefined	Center the controls vertically within the screen form.
ChangePrefix	undefined	Change the prefix applied to the variables generated by the screen.
CharBackTab	**Shift+Tab**	Move the cursor back one tab stop.
CharLeft	**Left**	Move the cursor one character to the left.
CharLeftExtend	**Shift+Left**	Select the text one character to the left.
CharRight	**Right**	Move the cursor one character to the right.
CharRightExtend	**Shift+Right**	Select the text one character to the right.
CharacterScreenImportUtility	undefined	Prepare for character screen import.
CharTab	**Tab**	Move the cursor forward one tab stop.
ClearParagraph	undefined	In the IO Handling tab of the File Designer, delete the paragraph name for the currently selected user-defined paragraph.
CodeTemplate	undefined	Open the Code Template list box.
CommentBlock	undefined	Apply comment symbols to the selected block of text.

Command	**Shortcut Key**	**Description**
ControlHeightDecrease	undefined	Decrease the height of the selected control by one pixel.
ControlHeightDecreaseGrid	undefined	Decrease the height of the selected control by one grid cell.
ControlHeightIncrease	undefined	Increase the height of the selected control by one pixel.
ControlHeightIncreaseGrid	undefined	Increase the height of the selected control by one grid cell.
ControlMoveDown	undefined	Move the selected control(s) down one pixel.
ControlMoveDownGrid	undefined	Move the selected control(s) down one grid cell.
ControlMoveLeft	undefined	Move the selected control(s) left one pixel.
ControlMoveLeftGrid	undefined	Move the selected control(s) left one grid cell.
ControlMoveRight	undefined	Move the selected control(s) right one pixel.
ControlMoveRightGrid	undefined	Move the selected control(s) right one grid cell.
ControlMoveUp	undefined	Move the selected control(s) up one pixel.
ControlMoveUpGrid	undefined	Move the selected control(s) up one grid cell.
ControlWidthDecrease	undefined	Decrease the width of the selected control(s) by one pixel.

Command	**Shortcut Key**	**Description**
ControlWidthDecreaseGrid	undefined	Decrease the width of the selected control(s) by one grid cell.
ControlWidthIncrease	undefined	Increase the width of the selected control(s) by one pixel.
ControlWidthIncreaseGrid	undefined	Increase the width of the selected control(s) by one grid cell.
Copy	**Ctrl+C**	Copy the selection to the clipboard.
CustomColor	undefined	Customize the basic set of colors.
Cut	**Ctrl+X**	Cut the selection and move it to the clipboard.
DebugBreakExecution	undefined	Terminate the debugging session and the runtime.
DebugBreakpointClearAll	undefined	Remove all breakpoints.
DebugBreakpointDisable	undefined	Disable the breakpoint on the cursor line.
DebugBreakpointManagement	undefined	Generate the Breakpoints dialog box.
DebugBreakpointToggle	**F9**	Toggle the breakpoint on the cursor line.
DebugGo	**F8**	Start the debugger or, if the debugger is already started, execute the program in Debug Mode.
DebugHexadecimalDisplay	undefined	Toggle hexadecimal display of data item values in the debugger.
DebugInterrupt	undefined	Interrupt the execution of the application, and return control to the debugger.

Command	**Shortcut Key**	**Description**
DebugMonitorClear	undefined	Clear the selected variable from the debug Watch window.
DebugMonitorClearAll	undefined	Clear all variables from the debug Watch window.
DebugMonitorSet	undefined	Set a monitor on a variable in the debug Watch window.
DebugQuickWatch	undefined	Open the Quick Watch window.
DebugRecordScript	undefined	Record a script of keyboard input and menu selections in a debugging session.
DebugRestart	undefined	Restart the debugging session at the beginning.
DebugRunScript	undefined	Run a recorded script of keyboard input and menu selections in a debugging session.
DebugRunToCursor	undefined	Debug the program from a temporary breakpoint at the cursor line.
DebugSkipToLine	undefined	Debug the program from the current cursor location.
DebugStepInto	**F11**	Debug the next statement.
DebugStepOutParagraph	undefined	Debug the program until the current paragraph returns to the point from which it was performed.
DebugStepOutProgram	undefined	Debugging the program until it exits to its calling program.
DebugStepOver	**F10**	Debug the program after stepping over the next statement.

Command	Shortcut Key	Description
DebugStopDebugging	**F12**	Stop the debugger, and continue execution of the program.
DebugStopRecordScript	undefined	Stop the recording of a script of keyboard input and menu selections in a debugging session.
DebugThreads	undefined	Debug program threads.
DebugTraceFiles	undefined	Enable the Trace File function.
DebugTraceFlush	undefined	Enable the Trace Flush function.
DebugTraceGeneral	undefined	Enable the Trace General function.
DebugTraceLevels	undefined	Generate the Trace Level dialog box.
DebugTraceParagraphs	undefined	Enable the Trace Paragraphs function.
DebugTraceScreens	undefined	Enable the Trace Screens function.
DebugTraceShowInfo	undefined	Display debugger trace information in the Output window.
Delete	**Delete**	Delete the selection.
DeleteAllKey	undefined	Delete all the keys in the File Designer’s key list.
DeleteBack	**Backspace**	Delete the selected text or, if no selection, the character to the left of the cursor.
DeleteFromDisk	undefined	Remove the selected file from the project and place it in the Windows Recycle Bin.

Command	**Shortcut Key**	**Description**
DeleteLine	undefined	Delete the line on which the cursor is positioned.
DockingComponentBar	undefined	Dock the Screen Component Toolbox.
DockingProjectManager	undefined	Dock the Workspace window.
DocumentEnd	**Ctrl+End**	Move the cursor to the end of the document.
DocumentEndExtend	**Ctrl+Shift+End**	Select the text from the current cursor position to the end of the document.
DocumentStart	**Ctrl+Home**	Move the cursor to the beginning of the file.
DocumentStartExtend	**Ctrl+Shift+Home**	Select the text from the current cursor position to the start of the document.
DragAndDrop	undefined	Toggle the Drag-and-Drop interface in the Screen and Report Designers.
ExternalParagraph	undefined	If a program icon is selected in the Structure view, open the External Paragraph list.
ExternalVariable	undefined	If a program icon is selected in the Structure view, open the External Variable list.
FileClose	**Ctrl+Q**	Close the file, and remove it from the development area.
FileCloseWorkspace	undefined	Close the workspace, and remove it from the Workspace window.
FileOpen	**Ctrl+O**	Open an existing source file in the development area.

Command	**Shortcut Key**	**Description**
FileOpenWorkspace	undefined	Open an existing workspace in the Workspace window.
FilePageSetup	**Ctrl+M**	Generate the Page Setup dialog box.
FilePrint	**Ctrl+P**	Print the active file.
FilePrintPreview	undefined	Display full pages in print preview.
FilePrintSetup	**Ctrl+K**	Generate the Print Setup dialog box.
FileSave	**Ctrl+S**	Save the active file.
FileSaveAll	undefined	Save all open files to disk.
FileSaveAs	undefined	Generate the Save As dialog box.
FileSaveWorkspace	undefined	Save the current workspace.
Find	undefined	Find the specified text in the active file.
FindInFiles	undefined	Find the specified text in the workspace or other designated directory.
FindInObjects	undefined	Find the specified text in the Property window and graphical designers.
FindInObjectsPopup	undefined	Open the Find in Objects interface.
FindNext	undefined	Find the next occurrence of the specified text.
FindPrev	undefined	Find the previous occurrence of the specified text.
FindReplace	undefined	Find the specified text and replace it with another designated string.

Command	**Shortcut Key**	**Description**
FindScope	undefined	Find the effective range of a selected variable.
Generate	undefined	Generate code for the program element in the active window.
GenerateSTFDocument	undefined	Generate an STF (Standard Text Format) description of the screen.
GenerateWTFDocument	undefined	Generate a WTF (report template file) for the selected report form.
GotoLine	undefined	Move the cursor to the specified line of code.
Help	**F1**	Provide context sensitive help. If the context is not recognized, display the online help directory.
HelpAcuBench	undefined	Display the AcuBench documentation.
HelpGTManual	undefined	Display the four-book ACUCOBOL-GT documentation set.
HelpRuntimeManual	undefined	Display the HTML version of the *Runtime Manual*.
HideComponentBar	undefined	Hide the Screen and/or Report Component Toolbox.
HideProjectManager	undefined	Hide the Workspace window.
Home	**Home**	Move the cursor to the start of the current line.
HomeExtend	**Shift+Home**	Select the text from the current cursor position in a line to the start of that line.

Command	**Shortcut Key**	**Description**
ImportCopyFile	undefined	Import a COPY file into the specified File Designer file descriptor.
IndentToNext	undefined	Indent the selected line to match the indentation of the next line
IndentToPrevious	undefined	Indent the selected line to match the indentation of the previous line.
InsertBefore	undefined	Insert an item before the selected item in the Menu or Tree View Designer.
InsertFileContent	undefined	Insert an existing file at the cursor position.
InsertMode	**Insert**	Toggle insert mode in the Code Editor.
InsertSeparatorBefore	undefined	Insert a separator before the currently selected menu item in the Menu Designer.
InsertSubitem	undefined	Insert a child item below the selected item in the Menu or Tree View Designer.
InsertTabCtrlPage	undefined	Insert a new page in the tab control.
LayoutAlignBottom	undefined	Align the bottom edges of the selected controls relative to the dominant control.
LayoutAlignLeft	undefined	Align the left edges of the selected controls relative to the dominant control.
LayoutAlignRight	undefined	Align the right edges of the selected controls relative to the dominant control.

Command	**Shortcut Key**	**Description**
LayoutAlignTop	undefined	Align the top edges of the selected controls relative to the dominant control.
LayoutOrderSendToFront	undefined	If multiple controls overlap, bring the selected control to the foreground.
LayoutOrderSendToBack	undefined	If multiple controls overlap, send the selected control to the back.
LayoutSizeToContent	undefined	Resize the selected control(s) to fit the title text.
LayoutSpaceEvenlyAcross	undefined	Evenly space the selected controls horizontally.
LayoutSpaceEvenlyDown	undefined	Evenly space the selected controls vertically.
LayoutTabOrder	undefined	Set the order in which your controls appear in the Screen Section.
LineDown	**Down**	Move the cursor down one line.
LineDownExtend	**Shift+Down**	Select the text from the current cursor position down one line.
LineEnd	**End**	Move the cursor to the end of the current line.
LineEndExtend	**Shift+End**	Select the text from the current cursor position to the end of the line.
LineUp	**Up**	Move the cursor up one line.
LineUpExtend	**Shift+Up**	Select the text from the current cursor position up one line.

Command	**Shortcut Key**	**Description**
LinkCopyFile	undefined	Link to a specified file descriptor from the File Designer.
ListConstants	undefined	Browse all level -78 constant names in the active file.
ListParagraphs	undefined	Browse all paragraph names in the active file.
ListVariables	undefined	Browse all variable names in the active file.
ListCopyFiles	undefined	Browse all COPY file names in the active file.
ListView	undefined	Display Screen Component Toolbox controls in a list.
LockControls	undefined	Lock/unlock the position of a control on a screen.
MakeSameHeight	undefined	Resize the selected controls to the same height.
MakeSameSize	undefined	Resize the selected controls to the same size.
MakeSameWidth	undefined	Resize the selected controls to the same width.
MakeXFDFile	undefined	Create an XFD for the selected Data Layout.
ModifyKey	undefined	Change a key in the specified file descriptor in the File Designer.
NativeCode	undefined	Open the Translate Portable Object Code to Native Code dialog for the selected object file.
New	**Ctrl+N**	Create a new project, program, screen, source file, or file descriptor.
NewDataSet	undefined	Create a new data set.

Command	**Shortcut Key**	**Description**
NewFD	undefined	Create a new data layout (".dlt") file.
NewLine	undefined	Insert a new line above the current cursor line.
NewProgram	undefined	Create a new AcuBench program.
NewScreen	undefined	Create a new screen.
NextControl	undefined	Select the next control in the tab order on the screen.
NextTabCtrlPage	undefined	Select the next page in the tab control.
NextTabPosition	undefined	Move the cursor to the next ANSI format area.
OpenCopyFile	undefined	Open the designated COPY file.
OpenFDEventEditor	undefined	Open the Event Editor for the data layout file selected in the Data view.
OpenProject	undefined	Add an existing project (saved as a ".pjf") to the current workspace.
OutputClear	undefined	Clear the contents of the Output window.
OutputCompileOptions	undefined	If a source file is selected in the File view, show the compiler options for that file in the Output window.
OutputFindInObjects	undefined	
OutputHistory	undefined	If the Output window is selected, toggle the History option (to append or overwrite existing output information).

Command	**Shortcut Key**	**Description**
OutputPreviousErrorLine	undefined	Position the cursor in source code at the location of the next previous compilation error listed in the Output window.
OutputNextErrorLine	undefined	Position the cursor in source code at the location of the next compilation error listed in the Output window.
PageDown	undefined	Move the cursor down one page.
PageDownExtend	undefined	Select the text from the current cursor position down one page.
PageUp	undefined	Move the cursor up one page.
PageUpExtend	undefined	Select the text from the current cursor position up one page.
Paste	**Ctrl+V**	Paste the selection from the clipboard to the active file.
PreviousControl	undefined	Select the previous control on the screen form.
PreviousTabCtrlPage	undefined	Select the previous page in the tab control.
PreviousTabPosition	**Ctrl+Shift+Tab**	Move the cursor to the previous ANSI format area.
ProgramCompileOptions	undefined	Open the Compile tab of the Project Settings window for the selected program or project.
ProjectAddRemoveFiles	undefined	Add or remove files from a project.
ProjectAssociateFDSL	undefined	Associate the added FD/SL files with the project.

Command	**Shortcut Key**	**Description**
ProjectCreateAlias	undefined	When the Create Alias option is available, open the Alias dialog.
ProjectEmptyFolder	undefined	Empty the specified project folder.
ProjectNewFolder	undefined	Create a new project folder.
ProjectOpen	undefined	Open the selected project.
ProjectProperty	undefined	Generate the Project Properties dialog box.
ProjectSetActiveMode	undefined	Select the appropriate mode for your project.
ProjectSetting	undefined	Generate the Project Settings dialog box.
Property	undefined	Generate the Properties dialog for the selected folder or file.
PropertyFont	undefined	Choose the Property window text font.
RebuildWorkspace	undefined	Recompile all files in the current workspace.
RebuildProject	undefined	Rebuild the project.
Redo	**Ctrl+Y**	Reverse the most recent Undo operation.
ReferencedDataFiles	undefined	In the Data Set Designer, open the Data Set Member Files dialog.
RefreshDataFile	undefined	In the Data view, reload the selected data layout file.
RefreshAllDataFiles	undefined	In the Data view, reload all data layout files in the selected project.

Command	**Shortcut Key**	**Description**
RegenerateWorkspace	undefined	Regenerate all programs that have a program structure file, as well as all data layout files, within the workspace.
ReloadCopyFile	undefined	In the Working Storage or Linkage Editor, reload the selected linked COPY file.
RemoveCopyFile	undefined	In the Working Storage or Linkage Editor, remove the selected linked COPY file.
Reparse	undefined	Reparse the current source file.
ReparseAll	undefined	Reparse all the workspace's source files.
ReplaceInFiles	undefined	Perform a search and replace operation in the specified directories.
ResetXFDName	undefined	
RptSectionController	undefined	Open the Section Controller interface in the Report Writer.
SaveProject	undefined	Save the selected project in a ".pjf" file.
ScreenImportUtility	undefined	Import the Screen Section as a new program.
ScrollDown	undefined	Scroll the Code Editor window down one line.
ScrollUp	undefined	Scroll the Code Editor window up one line.
SelectAll	**Ctrl+A**	Select all the elements in the active file.
SelectionCapitalize	undefined	Capitalize the selected text.
SelectionLowercase	undefined	Lowercase the selected text.

Command	**Shortcut Key**	**Description**
SelectionUppercase	undefined	Uppercase the selected text.
SequenceNumber	undefined	Generate the Sequence Number dialog box.
SetDefaultXFD	undefined	On the XFD tab of the File Designer, return the XFD to its default settings.
SyntaxCheck	undefined	Perform a syntax check in the Working-Storage, Linkage, or File Designer.
ToggleAlignToolbar	undefined	Toggle the display of the Align toolbar.
ToggleBookmarkPane	undefined	Toggle the display of the Bookmark pane.
ToggleScreenComponentBar	undefined	Toggle the display of the Screen Component Toolbox.
ToggleDebugToolbar	undefined	Toggle the display of the Debug toolbar.
ToggleEditorToolbar	undefined	Toggle the display of the Editor toolbar.
ToggleFullPathName	undefined	Toggle the display of the full pathname in the window title bar.
ToggleGrid	undefined	Toggle the display of the grid in a Screen Designer window.
ToggleGuide	undefined	Toggle the display of the Screen Designer’s control positioning guides.
ToggleLaunchToolbar	undefined	Toggle the display of the Launch toolbar.
ToggleLineNumberPane	undefined	Toggle the display of the Line Number pane.

Command	**Shortcut Key**	**Description**
ToggleMemoryWindow	undefined	Toggle the display of the Memory debugger window.
ToggleOutputWindow	undefined	Toggle the display of the Output window.
ToggleProjectManager	undefined	Toggle the display of the Workspace window.
ToggleProjectToolbar	undefined	Toggle the display of the Project toolbar.
TogglePropertyWindow	undefined	Toggle the display of the Property window.
ToggleReportComponentBar	undefined	Toggle Report Component Toolbox display.
ToggleRulerBar	undefined	Toggle the display of the ruler bar.
ToggleStackInfoWindow	undefined	Toggle the display of the Call Stack debugger window.
ToggleStandardToolbar	undefined	Toggle the display of the Standard toolbar.
ToggleStatusBar	**Ctrl+Shift+S**	Toggle the display of the status bar.
ToggleThreadWindow	undefined	Display the debugger Thread dialog box.
ToggleWatchWindow	undefined	Toggle the display of the debugger Watch window.
ToolsCustomize	undefined	Customize the toolbar settings.
ToolsOptions	undefined	Generate the Tools/Options dialog box.
TransposeSourceFormat	undefined	Transpose the document display format between ANSI and terminal formats.

Command	**Shortcut Key**	**Description**
UncommentBlock	undefined	Remove comment symbols from the selected text.
Undo	**Ctrl+Z**	Reverse previous operations.
UtilityAcu4glFD	undefined	Launch the **acu4glfd** utility.
UtilityAxdefgen	undefined	Launch the ACUCOBOL-GT utility **axdefgen**.
UtilityNetdefgen	undefined	Launch the ACUCOBOL-GT utility **netdefgen**.
UtilityCblUtil	undefined	Launch the ACUCOBOL-GT utility **cblutil**.
UtilityLogUtil	undefined	Launch the ACUCOBOL-GT utility **logutil**.
UtilityVio	undefined	Launch the ACUCOBOL-GT utility **vio**.
UtilityVioProject	undefined	With a project icon selected, launch **vio**.
UtilityVutil	undefined	Launch the ACUCOBOL-GT utility **vutil**.
Utility Xml2FD	undefined	Launch the ACUCOBOL-GT utility **xml2fd**.
VerbBlockMatch	**Ctrl+Shift+M**	Move to the opposite end of a verb block command statement.
ViewPreprocessed	undefined	View the ".asq" file created when the selected file is compiled with a preprocessor ("-P") flag.

Command	**Shortcut Key**	**Description**
ViewFDFile	undefined	Display the selected ".fd" file.
ViewLinkageFile	undefined	Display the selected Linkage Section (".lks") file.
ViewMenuFile	undefined	Dpslay the selected menu (".mnu") COPY file.
ViewParagraphFile	undefined	Display the selected event paragraph (".evt") file.
ViewProcedureFile	undefined	Display the selected Procedure Division (".prd") file.
ViewScreenFile	undefined	Display the selected screen (".scr") file.
ViewSLFile	undefined	Display the selected ".sl" file.
ViewSourceFile	undefined	Display the selected source (".cbl") file.
ViewWorkingFile	undefined	Display the selected Working-Storage (".wrk") file.
WindowArrangeIcons	undefined	Arrange the caption bars of minimized windows at the bottom of the screen.
WindowCascade	undefined	Arrange open windows in a cascading format.
WindowCloseAll	undefined	Close all open windows.
WindowNextPane	**F6**	Activate the next pane in a split window.
WindowPreviousPane	**Shift+F6**	Activate the previous pane in a split window.
WindowSplit	undefined	Split the open window.
WindowTileHorz	undefined	Tile the open windows horizontally.

Command	**Shortcut Key**	**Description**
WindowTileVert	undefined	Tile the open windows vertically.
WindowWindowsList	undefined	Generate the Windows list box showing all open windows.
WordLeft	**Ctrl+Left**	Move the cursor back one word.
WordLeftExtend	**Ctrl+Shift+Left**	Select the text from the current cursor position back one word.
WordBackDelete	undefined	Delete the word to the left of the current cursor position.
WordDelete	undefined	Delete the word to the right of the current cursor position.
WordRight	**Ctrl+Right**	Move the cursor forward one word.
WordRightExtend	**Ctrl+Shift+Right**	Select the text from the current cursor position forward one word.
XML2FD	undefined	In the Data view, with a project icon selected, launches the **xml2fd** utility.

23.3 Code Editor: Default Keyboard Shortcuts

Remember that some commands can be assigned keystrokes at both the Main level and the Code Editor level. This means that some common commands, such as Cut and Paste, are shown as undefined in the list that follows, even though a keyboard shortcut has been assigned to them at the Main level.

Command	**Shortcut Key**	**Description**
BookmarkClearAll	undefined	Clear all bookmarks.

Command	**Shortcut Key**	**Description**
BookmarkNext	**Ctrl+Shift+N**	Move the cursor to the line containing the next bookmark.
BookmarkPrev	**Ctrl+Shift+P**	Move the cursor to the line containing the previous bookmark.
BookmarkToggle	**Ctrl+T**	Toggle the display of a bookmark on the current line.
CharBackTab	**Shift+Tab**	Move the cursor back one tab stop.
CharLeft	**Left Arrow**	Move the cursor one character to the left.
CharLeftExtend	**Shift+Left Arrow**	Select the text one character to the left.
CharRight	**Right Arrow**	Move the cursor one character to the right.
CharRightExtend	**Shift+Right Arrow**	Select the text one character to the right.
CharTab	**Tab**	Move the cursor forward one tab stop.
CodeTemplate	**Ctrl+J**	Open the code template list box.
CommentBlock	**Ctrl+Shift+C**	Apply comment symbols to the selected block of text.
Copy	undefined	Copy the selected text to the clipboard.
Cut	undefined	Cut the selected text and move it to the clipboard.
Delete	undefined	Delete the selected text.
DeleteBack	**Backspace**	Delete the selected text, or if no selection, the character to the left of the cursor.

Command	**Shortcut Key**	**Description**
DeleteLine	**Ctrl+Shift+Z**	Delete the line on which the cursor is positioned.
DocumentEnd	**Ctrl+End**	Move the cursor to the end of the document.
DocumentEndExtend	**Ctrl+Shift+End**	Select the text from the current cursor position to the end of the document.
DocumentStart	**Ctrl+Home**	Move the cursor to the beginning of the file.
DocumentStartExtend	**Ctrl+Shift+Home**	Select the text from the current cursor position to the start of the document.
FilePrint	undefined	Print the document.
FilePrintPreview	undefined	Display full pages in print preview.
Find	**Ctrl+F**	Find the specified text in the active file.
FindNext	**F3**	Find the next occurrence of the specified text in the active file.
FindPrev	**Shift+F3**	Find the previous occurrence of the specified text in the active file.
FindReplace	**Ctrl+H**	Find the specified text and replace it with another designated string.
FindScope	**Ctrl+Shift+V**	Find the effective range of a selected variable.
GotoLine	undefined	Move the cursor to the specified line of code.
Home	**Home**	Move the cursor to the start of the current line.

Command	**Shortcut Key**	**Description**
HomeExtend	**Shift+Home**	Select the text from the current cursor position in a line to the start of that line.
IndentToNext	undefined	Indent the selected cursor line to match the indentation of the next line.
IndentToPrevious	undefined	Indent the selected cursor line to match the indentation of the previous line.
InsertFileContent	undefined	Insert an existing document at the cursor position.
InsertMode	**Insert**	Toggle insert mode in the Code Editor.
LineDown	**Down Arrow**	Move the cursor down one line.
LineDownExtend	**Shift+Down Arrow**	Select the text from the current cursor position down one line.
LineEnd	**End**	Move the cursor to the end of the current line.
LineEndExtend	**Shift+End**	Select the text from the current cursor position to the end of the line.
LineUp	**Up Arrow**	Move the cursor up one line.
LineUpExtend	**Shift+Up Arrow**	Select the text from the current cursor position up one line.
NextTabPosition	**Ctrl+Tab**	Move the cursor to the next ANSI format area.
NewLine	**Ctrl+Shift+I**	Insert a new line above the current cursor line.

Command	**Shortcut Key**	**Description**
OpenCopyFile	undefined	Open the designated COPY file.
PageDown	**Page Down**	Move the cursor down one page.
PageDownExtend	**Shift+Page Down**	Select the text from the current cursor position down one page.
PageUp	**Page Up**	Move the cursor up one page.
PageUpExtend	**Shift+Page Up**	Select the text from the current cursor position up one page.
Paste	undefined	Paste the selected text from the clipboard to the active file.
PreviousTabPosition	**Ctrl+Shift+Tab**	Move the cursor to the previous ANSI format area.
Redo	undefined	Reverse the most recent undo operation.
ScrollDown	**Ctrl+Down**	Scroll the Code Editor window down one line.
ScrollUp	**Ctrl+Up**	Scroll the Code Editor window up one line.
SelectAll	undefined	Select all the text in the active file.
SelectionCapitalize	**Ctrl+G**	Capitalize the selected text.
SelectionLowercase	**Ctrl+U**	Lowercase the selected text.
SelectionUppercase	**Ctrl+Shift+U**	Uppercase the selected text.
SequenceNumber	undefined	Generate the Sequence Number dialog box.

Command	**Shortcut Key**	**Description**
TransposeSourceFormat	undefined	Transpose the document display format between ANSI and terminal formats.
UncommentBlock	undefined	Remove comment symbols from the selected text.
Undo	undefined	Reverse previous operations.
VerbBlockMatch	**Ctrl+Shift+M**	Move to the opposite end of a verb block command statement.
WordBackDelete	**Ctrl+B**	Delete the word to the left of the current cursor position.
WordLeft	**Ctrl+Left**	Move the cursor back one word.
WordLeftExtend	**Ctrl+Shift+Left**	Select the text from the current cursor position back one word.
WordDelete	**Ctrl+D**	Delete the work to the right of the current cursor position.
WordRight	**Ctrl+Right**	Move the cursor forward one word.
WordRightExtend	**Ctrl+Shift+Right**	Select the text from the current cursor position forward one word.

23.4 Screen Designer: Default Keyboard Shortcuts

Remember that some commands can be assigned keystrokes at both the Main level and the Screen Designer level. This means that some common commands, such as Cut and Paste, are shown as undefined in the list that follows, even though a keyboard shortcut has been assigned to them at the Main level.

A	C	D	F	G	I	L	M	N	P	R	S

Command	**Shortcut Key**	**Description**
ActiveXProperty	undefined	Display the ActiveX control's properties.
AdjacentHorizontal	undefined	Align controls to be adjacent horizontally.
AdjacentVertical	undefined	Align controls to be adjacent vertically.
AlignToGrid	undefined	Align control(s) to grid.
CenterHorizontal	undefined	Center the controls horizontally within the screen form.
CenterVertical	undefined	Center the controls vertically within the screen form.
ChangePrefix	undefined	Change the prefix applied to variables generated by the screen.
Copy	undefined	Copy the selected screen element to the clipboard.
ControlHeightDecrease	**Shift+Up Arrow**	Decrease the height of the selected control by one pixel.
ControlHeightDecreaseGrid	**Ctrl+Shift+Up Arrow**	Decrease the height of the selected control by one grid cell.

Command	**Shortcut Key**	**Description**
ControlHeightIncrease	**Shift+Down Arrow**	Increase the height of the selected control by one pixel.
ControlHeightIncreaseGrid	**Ctrl+Shift+Down Arrow**	Increase the height of the selected control by one grid cell.
ControlMoveDown	**Down Arrow**	Move the selected control(s) down one pixel.
ControlMoveDownGrid	**Ctrl+Down Arrow**	Move the selected control(s) down one grid cell.
ControlMoveLeft	**Left Arrow**	Move the selected control(s) left one pixel.
ControlMoveLeftGrid	**Ctrl+Left Arrow**	Move the selected control(s) left one grid cell.
ControlMoveRight	**Right Arrow**	Move the selected control(s) right one pixel.
ControlMoveRightGrid	**Ctrl+Right Arrow**	Move the selected control(s) right one grid cell.
ControlMoveUp	**Up Arrow**	Move the selected control(s) up one pixel.
ControlMoveUpGrid	**Ctrl+Up Arrow**	Move the selected control(s) up one grid cell.
ControlWidthDecrease	**Shift+Left Arrow**	Decrease the width of the selected control(s) by one pixel.
ControlWidthDecreaseGrid	**Ctrl+Shift+Left Arrow**	Decrease the width of the selected control(s) by one grid cell.
ControlWidthIncrease	**Shift+Right Arrow**	Increase the width of the selected control(s) by one pixel.

Command	**Shortcut Key**	**Description**
ControlWidthIncreaseGrid	**Ctrl+Shift+Right Arrow**	Increase the width of the selected control(s) by one grid cell.
CustomColor	undefined	Customize the basic set of colors.
Cut	undefined	Cut the selected screen element and move it to the clipboard.
Delete	undefined	Delete the selected screen element.
FilePrint	undefined	Print the active screen.
FilePrintPreview	undefined	Display full screens in print preview.
GenerateSTFDocument	undefined	Save the screen as an ".stf" file.
InsertBefore	**Insert**	Insert an item before the selected item in the Menu or Tree View designer.
InsertSeparaterBefore	undefined	
InsertSubitem	**Ctrl+Insert**	Insert a child item below the selected item in the Menu or Tree View Designer.
InsertTabCtrlPage	undefined	Insert a new page in the tab control.
LayoutAlignBottom	**Alt+Down**	Align the bottom edges of the selected controls relative to the dominant control.
LayoutAlignLeft	**Alt+Left**	Align the left edges of the selected controls relative to the dominant control.

Command	**Shortcut Key**	**Description**
LayoutAlignRight	**Alt+Right**	Align the right edges of the selected controls relative to the dominant control.
LayoutAlignTop	**Alt+Up**	Align the top edges of the selected controls relative to the dominant control.
LayoutOrderSendToFront	undefined	If multiple controls overlap, bring the selected control to the foreground.
LayoutOrderSentToBack	undefined	If multiple controls overlap, send the selected control to the back.
LayoutSizeToContent	**Shift+F7**	Resize the selected control(s) to fit the title text.
LayoutSpaceEvenlyAcross	undefined	Evenly space the selected controls horizontally.
LayoutSpaceEvenlyDown	undefined	Evenly the space selected controls vertically.
LayoutTabOrder	**Ctrl+D**	Set the order in which your controls appear in the Screen Section.
LockControls	**Ctrl+L**	Lock/unlock the position of a control on a screen.
MakeSameHeight	undefined	Resize the selected controls to the same height.
MakeSameSize	undefined	Resize the selected controls to the same size.
MakeSameWidth	undefined	Resize the selected controls to the same width.
NextControl	**Tab**	Select the next control in the tab order on the screen.

Command	**Shortcut Key**	**Description**
NextTabCtrlPage	undefined	Select the next page in the tab control.
Paste	undefined	Paste the selected screen element from the clipboard to the active screen.
PreviousControl	**Shift+Tab**	Select the previous control on the screen form.
PreviousTabCtrlPage	undefined	Select the previous page in the tab control.
Redo	undefined	Reverse the most recent undo operation.
ToggleGrid	**Ctrl+G**	Toggle the display of the grid in a Screen Designer window.
ToggleGuide	undefined	Toggle the display of the Screen Designer's control positioning guides.
Undo	undefined	Reverse previous operations.
SelectAll	undefined	Select all the elements in the active screen.

24 Bringing Existing Code Into AcuBench

Key Topics

Introduction 24-2
Importing Data Definitions 24-3
Creating a PSF for an Existing Program 24-4
Adding Existing Code to an AcuBench Program 24-8
Updating a Character-based Screen 24-11
Importing a Graphical Screen 24-30

24.1 Introduction

If you want to use AcuBench's code generation tools with your existing COBOL source code, there are three basic options available. You can:

- Create a new AcuBench program and use it to generate screen or report code, then copy the generated code into your existing source file.
- Create a program structure file for your existing source file.
- Create a new AcuBench program and use the graphical designers to copy and paste code from your existing source files into the new program.

Each option has its own advantages and considerations.

- If you choose to use an AcuBench template program to generate code that you then place into your existing source files, you retain your familiar code format and strict control over the contents of your source files. You lose, however, the flexibility of using the drag-and-drop design interfaces to make quick changes to your program.
- If you choose to create a program structure file for your existing source file, you gain access to some of the AcuBench code generation facilities while retaining most of the familiar code structure of your existing file. Because AcuBench has a limited ability to parse and understand your existing code, however, not all of the workbench's automation tools are available for your use.
- If you choose to bring your existing source code into a new AcuBench program, you gain access to all of the automated code generation tools that the workbench provides. This process may, however, be time consuming and requires adaptation to AcuBench's formats and processes.

Whichever option you choose, keep in mind that to make best use of the tools that AcuBench provides, converting the source code is only one part of the process. By creating data layout files (".dlt") to define your data and data handling code for AcuBench, you greatly increase the utility and functionality of the AcuBench graphical designers.

24.2 Importing Data Definitions

As discussed in **Chapter 8**, there are several advantages to bringing your data file definitions—file descriptors and SELECT statements—into AcuBench. It can help to centralize maintenance of data definition and file-handling code and enable various automation tools, like the Drag and Drop interface used in screen and report design and the Data Set Designer, used to define data for use with individual programs.

The easiest way to bring existing data definitions into AcuBench is to first create COPY files corresponding to each file descriptor and SELECT statement. For each FD/SELECT pair:

- The two files should have the same base name.
- The FD file should have a ".fd" extension.
- The SELECT file should have a ".sl" extension.

When you have finished creating COPY files, open your AcuBench project and do the following:

1. In the File view, right-click the **FD** folder and select **Add/Remove Files**.
2. Use the navigation tree view in the top, right corner of the Add/Remove Files dialog to navigate to the directory containing your COPY files.
3. Select the files from the file list, then click the **Add** button. To add all of the files in the directory, navigate to the directory and click **Add All**.
4. When you have finished adding COPY files (which may reside in multiple directories), click **OK**.
5. Switch to the Data view, right-click the project node, and select **Associate FD/SL Files**.
6. In the window that opens, match the FD/SL pair and click **Associate**.
7. Continue this process until all FD/SL pairs have been associated and added to the list in the middle of the dialog, then click **OK**.

   A new data layout file (DLT) appears in the Data view for each FD/SL pair.

Once you have created data layout files from your existing file descriptors and SELECT statements, you can use the AcuBench File Designer to maintain your data descriptions and file handling code. For more information about the File Designer and its capabilities, please refer to **Chapter 8, "Chapter 8: Working with Data at the Project Level."**

24.3 Creating a PSF for an Existing Program

When you create a program structure file for an existing source file, AcuBench also adds code generation tags to your ".cbl" source file. When you generate your program, these tags are used to tell AcuBench where to generate various types of code (Working-Storage, Screen Section, and so on). You can determine which tags are included in your code through the Tools/Options/Code Generator/Program Tag interface. Use of code generation tags is discussed in **Chapter 4, section 4.6.2, "Program Tag Options."**

24.3.1 Prepare to Create the PSF

Before you perform the steps described in this chapter, make sure that AcuBench's code generation is set to generate code only between AcuBench program tags.

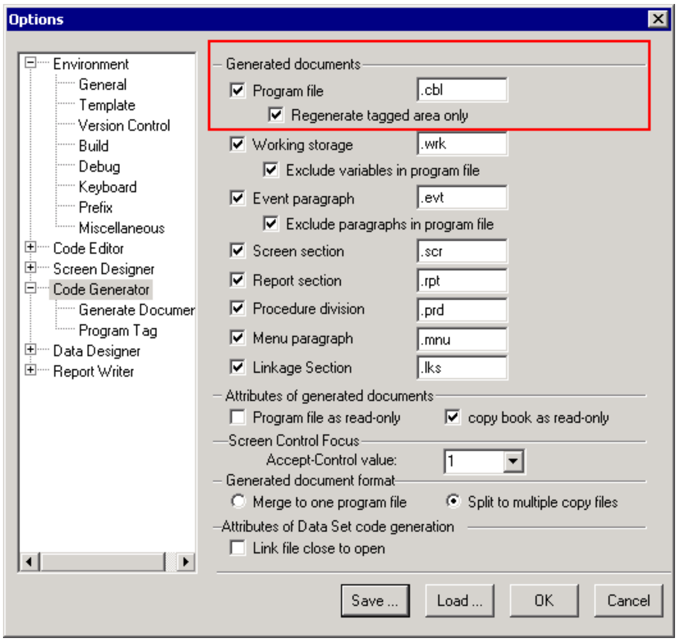
Caution: If you neglect this step, the first time you generate your program, all of the code outside the AcuBench tags—meaning all the code that you have written inside your source file—will be overwritten.

To ensure that only code between program tags is generated:

1. Expand the Tools menu and select **Options**.

2. In the Tools/Options interface, select the **Code Generator** tree item.

3. Make sure that the second check box from the top of the screen is marked, as shown here:



Options

Generated documents
- [✓] Program file .cbl
  - [✓] Regenerate tagged area only
- [✓] Working storage .wrk
  - [✓] Exclude variables in program file
- [✓] Event paragraph .evt
  - [✓] Exclude paragraphs in program file
- [✓] Screen section .scr
- [✓] Report section .rpt
- [✓] Procedure division .prd
- [✓] Menu paragraph .mnu
- [✓] Linkage Section .lks

Attributes of generated documents
- [ ] Program file as read-only
- [✓] copy book as read-only

Screen Control Focus
- Accept-Control value: 1

Generated document format
- ( ) Merge to one program file
- (●) Split to multiple copy files

Attributes of Data Set code generation
- [ ] Link file close to open

By default, this box is marked, but it is always a good idea to verify this setting before adding a new program to AcuBench.

24.3.2 Create the PSF

To create a program structure file for an existing source file and add the updated program to a project, use the following steps:

1. In the File view of the Workspace window, expand the project node to which you plan to add your program.
2. Right-click the project's Source folder and select **Add/Remove Files**.
3. Locate and select your source file(s) in the "Files to be added" section of the dialog.
4. Click the **Add** button and verify that the name of each selected source file appears in the "Files in project" list at the bottom of the dialog.

5. Make sure that the check box in the “Create .PSF” column of the “Files in Project” list is marked for each source file for which you want to add a program structure file.

6. Click **OK**. AcuBench creates a program structure file and inserts tags into the source file.

   When you switch to the Structure view, a new program node appears in your project. Note that the program icon is different than the icon used for new AcuBench programs that you create. This serves as a reminder that some of the workbench’s automation tools are not available with programs that have been imported into AcuBench in this manner.

24.3.3 Key-Status

By default, AcuBench generates Key-Status code for any program that contains a program structure file. If you already have Key-Status code defined in your program, you will need to take steps to avoid duplicate definitions.

The simplest option is to disable the AcuBench option that generates Key-Status code. To do this:

1. Right-click the program in the Structure view and select **Properties**.

2. On the Key Status tab of the Program Properties window, mark the **Do not generate CRT STATUS variable in .wrk** check box.

   This check box appears near the bottom of the dialog.

3. Click **OK** to save your changes.

Note that there are five key-status definitions that are required if you are using AcuBench-generated graphical screens. These are:

```
88 Exit-Pushed            VALUE 27.
88 Message-Received       VALUE 95.
88 Event-Occurred         VALUE 96.
88 Screen-No-Input-Field  VALUE 97.
88 Screen-Time-Out        VALUE 99.
```

If your Key-Status definition does not contain these items and you use the AcuBench Screen Designer to add screens to your program, you will receive errors at compile time.

If you prefer to use the AcuBench-generated Key-Status code, you can copy and paste your own Key-Status variable into the generated definition. To do this:

1. Right-click the program in the Structure view and select **Properties**.
2. On the Key Status tab of the Program Properties window, make sure that the **Do not generate CRT STATUS variable in .wrk** check box is not selected.

   This check box appears near the bottom of the dialog.
3. Use the **Import** or **Insert** button to add conditions and values to the Key-Status definition.

   The Import option allows you to read in items from a data file; the Insert option allows you to manually add items one at a time.
4. When you are finished, click **OK** to save your changes.

24.3.4 Import Program Elements into the PSF

Once you have created a program structure file for your source program, you can chose which elements of your program to leave as they are, and which to place into AcuBench's graphical interfaces. A first step might be to copy your Working-Storage code into the graphical Working Storage Designer. After performing this step, AcuBench stores your Working-Storage data in the program structure file, and you can view and modify Working-Storage items through the graphical design interface.

You may also want to add your existing screens to AcuBench so that you can maintain and update them using the graphical Screen Designer. The steps involved in this process can be found in **section 24.5, "Updating a Character-based Screen,"** and **section 24.6, "Importing a Graphical Screen."**

24.4 Adding Existing Code to an AcuBench Program

To preserve the greatest amount of flexibility and ensure the greatest long-term ease of maintenance, the best option is adding your existing source code to a new AcuBench program. This option provides you with access to all of the AcuBench code generation and automation tools, and after the initial investment of time, requires the least amount of manual intervention to maintain and add to your code going forward. If you are using AcuBench as part of a legacy modernization process that involves increased use of graphical user interfaces and/or graphical reports, the ease with which you can add and update graphical features to an AcuBench program will likely render this initial investment well worthwhile.

In preparation for this process, the first thing to understand is how AcuBench generates code, and specifically how AcuBench uses program tags (discussed in **section 4.6.2**) to determine where to generate code. These program tags act as cues, telling AcuBench where to place different sections of a COBOL program. They are also used to indicate insertion points, at which the code that you define in the AcuBench graphical designers is placed. To understand how to bring your existing code into a new AcuBench program, you must understand where to place your code, so that it can be integrated into the generated program.

Step 1: Define your data

As with every method of working with AcuBench, the process starts with data. When you bring your FD and SELECT statements into the File Designer, the IO Handling interface provides a central repository for storing the file handling code associated with a specific data file.

Specific methods for bringing your data definitions into AcuBench are discussed in **Chapter 8, "Chapter 8: Working with Data at the Project Level."**

Step 2: Create the program

Before creating the AcuBench program, think about how code will be generated for the program. Do you want code to be generated into a single source file or split into multiple COPY files (the default)? Which program tags will you need? Study the Code Generator section of the Tools/Options interface (described in **Chapter 4, section 4.6, "Setting Code Generator Options"**) and set the code generation rules that best apply to your practices.

Chapter 9, "Chapter 9: Working with Programs," provides an overview of the program creation process, including information about the program properties that you can customize and control.

Step 3: Define your data sets

Data sets are used to specify how the data defined in project-level data layout files are used by an individual program. When you define a data set, you determine what file handling code is generated (OPEN, READ, DELETE, CLOSE), which key (if applicable) is used to start the file, and so on. In addition, data defined in data sets is exposed to the various AcuBench graphical designers, making it easier to create screens and reports, for example.

The process of defining data sets is described in **Chapter 10, "Chapter 10: Working with Data at the Program Level."**

Step 4: Add your Working-Storage data

By default, once a program contains screens and reports, AcuBench performs an automatic check to verify that the variables used by those screens and reports have been defined within the program. If AcuBench cannot locate those data definitions, it adds them to the graphical Working-Storage Designer. To correct problems with duplicate names, it is a good idea to bring your existing Working-Storage data definitions into the graphical designer before adding screens or other code.

You can use either of the following methods to bring your Working-Storage code into AcuBench:

- Copy and paste the code in from your existing source.
- Place the Working-Storage code in a COPY file and use the **Link COPY File** or **Import COPY File** commands from within the designer.

The Working-Storage Designer interface is described in **Chapter 10, section 10.3, "Using the Working-Storage and Linkage Editors."**

Note that, by default, AcuBench generates Key-Status definitions for all generated programs. If your Working-Storage section contains Key-Status definitions, you can either disable the automatic generation of Key-Status code or add your definitions to the AcuBench interface used to generate this code. These Key-Status options are discussed in **section 24.3.3, "Key-Status."**

Step 5: Import or re-create your screens

The last two sections of this chapter discuss the process of importing existing character-based and graphical screens into AcuBench. These import utilities provide a starting point for working with existing screens in the graphical Screen Designer.

As you begin to work in the Screen Designer, keep in mind the following points:

- Although the Property window contains long lists of screen and control properties, most of these settings shown here are defaults that you will have no need to change.

- In most cases, when the Property sheet provides a way to associate a fixed value (Value property) with an item, it also provides a way to associate a variable (Value Variable property) with the item.

- The Event tab of the Property sheet serves two purposes. It helps you associate codes with screen events and exceptions, and provides access to insertion points in the generated source (Before Routine, After Routine, Before Initdata, After Initdata) where you can add your own code.

- The Event Editor provides additional insertion points for adding program code (Before Program, After Program) and file handling code (Declaratives, Before DeleteFile), accessed by selecting a program or data set from the **Control** drop-down list, then selecting an insertion point from the **Message** drop-down list.

The Screen Designer and Event Editor discussed in **Chapter 14, "Chapter 14: Working with Screens."**

Step 6: Add your code

As discussed in Step 5, the Event Editor is both used for event, exception, and termination procedures, and to access insertion points in the generated code. Each of these insertion points is associated with a program tag, making them relatively easy to identify when stepping through the code in the debugger.

It is difficult to offer general rules for how to determine which pieces of code go where. The better you understand AcuBench's code generation rules, and the program tags that the code generator uses to identify and place code within the code skeleton, the more straightforward the process will be.

24.5 Updating a Character-based Screen

To help you transform your character-based application into a graphical one, ACUCOBOL-GT includes a screen conversion tool known as the Character-to-GUI Wizard. The tool works by watching a running character application and, at a specified point, constructing an equivalent graphical screen. This screen is automatically imported into the AcuBench Screen Designer where you can make modifications to it as desired. You can then use AcuBench to produce a Screen section description of the graphical screen and integrate the new Screen section into your original program.

The Character-to-GUI Wizard is designed to work with any character screen that you have created using ACUCOBOL-GT syntax. This allows it to work with most existing character-based programs. Note that the wizard works with programs using either of ACUCOBOL-GT's two main screen handling techniques: the Screen section and in-line ACCEPT/DISPLAY statements.

When the wizard converts a character screen to a graphical one, each field in the original screen is created as either a label or an entry field. These controls are placed in the same positions as the corresponding fields and have the same size. Labels are given the same text as the fields from which they are created. The source data item, if any, is attached to the control in such a way that it becomes the corresponding "value" property when the control is viewed in the Screen Designer. Most other information, including color, is not copied. Color is not copied because most character color schemes do not look good under a graphical system.

The wizard uses special rules when deciding whether to make a field a label or an entry field. If you find that these conversion rules result in undesirable effects (for instance, screens with an overabundance of entry fields), you can try converting the screen using alternate rules. To do so, select the **Use Alternate Rules** check box on the Properties dialog that displays after the initial conversion. This causes the wizard to re-evaluate the original character screen using slightly different conversion heuristics. You can try both to see which gives better results.

24.5.1 Benefits and Restrictions

The main benefits of using the Character-to-GUI Wizard are the following:

- The conversion process is faster than manual techniques.
- The resulting screens are placed in AcuBench, ready to use the powerful facilities of the Screen Designer to modernize the screen further. This also provides a nice starting point for bringing the entire application into AcuBench.
- Because the graphical interface is derived from the character one, it retains the efficiencies that have been built into the character interface, and the application's user base will find the interface familiar and comfortable.
- The resulting program is an ACUCOBOL-GT program. This program can be easier to understand and maintain than those produced via some other conversion techniques that may involve rewriting the screens in another programming language or using an external tool.
- By focusing the tasks to accomplish, the wizard can make the programmer more productive. Replacing an application's user interface with a graphical one is a daunting task with many decisions to be made. Using the wizard, the task is more concrete: integrating a screen into a program.

- If you need to retain the character-based interface, the wizard's output works well with ACUCOBOL-GT's dual-interface features. This makes it easy for a single application to contain both a character and a graphical user interface.

- The wizard creates screens that refer to the original data items used to build the character screen. This makes integration of the graphical screen into the program easier.

The Character-to-GUI Wizard has some important limitations that you should also consider:

- The wizard is available under Windows only.

- Because it mimics the character interface, the resulting graphical user interface makes use of only labels and entry fields. The programmer can add more graphical elements right away or over time as needed, but they are not included as part of the initial conversion.

- The wizard only handles screen labels and entry fields. More advanced elements of a character application are not recognized by the wizard. These include menu bars, pop-up windows, and line drawing characters.

- The wizard uses some source code analysis, some runtime analysis, and some heuristics to decide what on the character screen should be labels and what should be entry fields. While these are usually correct, in some cases the wizard could make the wrong choice. The wizard contains a method for correcting this when it occurs.

- Spacing issues arise when you convert from a fixed-pitch font in the character application to a variable-pitch font in the graphical application. In addition, the boxes around the graphical entry fields occupy screen space that is not present in the character application. As a result, screen elements do not always fit correctly after the conversion. The wizard uses a general-purpose placement algorithm that works well for most screens, but does not ensure that screen fields always fit or look right. You may have to occasionally move or resize a screen field after the screen has been imported into the Screen Designer. You can choose to minimize this effect by using a fixed-pitch screen font and/or using unboxed entry fields. The wizard supports both of these options.

- Screens constructed via techniques other than using ACUCOBOL-GT syntax cannot be converted or convert poorly. This also applies to screens constructed by embedding terminal control codes directly in the displayed data by the application. The wizard is not aware of the effects of the terminal control codes and treats them as data instead. The resulting screens are generally not useful.
- The programmer is responsible for integrating the output of the wizard (a Screen section item) back into the program. The wizard tries to simplify this task by referring to the original data items and screen names as appropriate. However, in some cases, the data item may not be correct or even legal. For example, if you construct some portion of a screen out of data contained in a table, the resulting Screen section item could contain several identical table references for logically distinct items. (This would happen if the screen were constructed by processing the table in a loop.)
- Screens that are not treated by the application as a series of fields do not convert well. For example, if you have a word processor application where the body of the screen is managed by single-character ACCEPT/DISPLAY statements, the wizard tries to treat each character as a distinct field. The results are not useful.
- The wizard does not handle screens whose set of fields change based on other actions. Essentially, when you import you get an image of the screen as it is at that time. If the screen is dynamic, then that image expresses only one form of the screen.

24.5.2 Using the Character-to-GUI Wizard

The following steps summarize the use of the Character-to-GUI Wizard. The remainder of this section describes aspects of the process in more detail.

1. Get the character application running with ACUCOBOL-GT under Windows.
2. Compile the application with "-Zw".
3. From AcuBench, run the application with the "--char2gui" option.

4. Convert the desired screens by right-clicking within them and selecting **Build Graphical Screen**.
5. Edit properties as desired in the resulting Properties window.
6. Make any desired changes to the new screens in the AcuBench Screen Designer.
7. Generate a new Screen section definition from the Screen Designer.
8. Include this Screen section definition in your program and connect it to your program logic.

24.5.3 Running under Windows

To use the wizard, you must first get your program running using ACUCOBOL-GT on a Windows machine. The wizard uses both the compiler and runtime to do its job, so you must first get your program into a state where it can be run under Windows.

24.5.4 Setting Compiler Options

For best results, the program containing the screen to be converted should first be compiled with the ACUCOBOL-GT compiler using the "-Zw" option. This option, which is set through the AcuBench Project Settings dialog box or on the ACUCOBOL-GT command line, causes the compiler to insert "comments" into the compiled object code that the wizard will use to determine the names of relevant data items. You can use the wizard without first compiling with "-Zw", but the wizard is less likely to select the proper control type for each field and the resulting Screen section will not contain any data names.

When compiling, you can also select a mapping option to define how your data names appear in the screen description: in uppercase, lowercase, or mixed case. By default, uppercase data names are displayed. Note that the case transformation occurs only for characters in the ASCII character set (the English alphabet). For characters outside of the ASCII character set, no transformation is performed.

Use the following steps to set the “-Zw” compiler option:

1. Use the **Project/Settings** command to display the Project Settings dialog box. Choose the **Compiler** tab.
2. Select **Miscellaneous Options** in the catalog drop-down box. Check the “-Zw” option, “Prepare for use with the Screen Import Utility.”
3. Select **OK** to save the setting.
4. Select **Save as Default** to save the setting in the “default.pof” file. These settings are applied to new projects when they are created.

Set mapping options as follows:

1. Use the **Project/Settings** command to display the Project Settings dialog box. Select the **Compiler** tab.
2. Select **Mapping Options** in the catalog drop-down box.
3. Select the option that best fits your needs:

-Mu	Names are uppercase
-Ml	Names are lowercase
-Mm	Names are mixed case: upper for the first letter of each word, lower for the rest

4. Select **OK** to save the setting.
5. Select **Save as Default** to save the setting in the “default.pof” file. These settings are applied to new projects when they are created.

24.5.5 Setting the “--char2gui” Runtime Option

After the relevant programs are compiled, you need to execute them. Perform this step as if you were running the programs normally, except add the “--char2gui” option to the runtime command line in order to launch the Character-to-GUI Wizard along with your program. In AcuBench, you add the “--char2gui” option to the runtime through the Project Settings dialog box as follows:

1. Select the **Project/Settings** command to display the Project Settings dialog box. Choose the **Runtime** tab.

2. Select **Common Options** in the catalog drop-down box. Check the "--char2gui" option, "Import Character Screen File."

3. Select **OK** to save the setting.

4. Select **Save as Default** to save the setting in the "default.pof" file. These settings are applied to new projects when they are created.

Alternatively, to select the "--char2gui" option, you can right-click the program node in the File tab and select the **Character Screen Import** command from the pop-up menu. Note that only the Windows 32-bit runtime supports this option.

24.5.6 Screen Conversion

After you have compiled and executed the program, you are ready to convert your text-based screens into graphical ones. See "Tips and Techniques" in **section 24.5.11** for a list of things you may want to consider before beginning.

Note: Don't bother converting message screens, because these screens come from the Windows message handler rather than your application. The Windows runtime automatically displays messages in a Windows message box.

Perform the screen conversion operation as follows:

1. With the relevant program running, right-click in the screen you want to convert.

2. To "capture" the screen for conversion, select the **Build Graphical Screen** command from the resulting pop-up menu. At this point, a graphical version of the application screen is created based on the wizard's internal conversion rules. The new graphical screen is displayed along with a Properties dialog box. You can use the Properties dialog to make global changes to the window created by the

wizard. This is described in **section 24.5.7, "Editing Screen Properties."** You can also change the size of the graphical window by dragging its borders.

Note that the original application screen is still present, but it is disabled until you finish with the wizard.

3. Navigate to the next screen that you want to convert and repeat the process until you have converted each screen in the program. The results are saved together in a single text file named "import.out". If a file named "import.out" already exists in the current working directory, it is overwritten.

4. Terminate the program normally. The converted screens are represented in your AcuBench project in the workspace Structure view in a new program. From there, they can be moved, opened, and modified as needed. Multiple screens are opened "cascade" style in the Screen Designer.

24.5.7 Editing Screen Properties

When the Character-to-GUI Wizard first displays a new graphical screen, it also displays the Properties dialog box so you can make changes if desired.

The Properties dialog allows you to change the conversion rules that the wizard used when first creating your graphical screens. Changes to individual screen items are made using the Screen Designer in AcuBench after the wizard has finished.

The Properties dialog contains the following options:

Option	**Description**
Unboxed fields	Creates entry fields without boxes. The default is to use boxes. Unboxed entry fields are useful if, for example, you need the resulting screen to fit on a 640 x 480 display. Most of the time, however, you would not want to use this option as the resulting screen does not have a Windows look.
Layout	Lets you choose how the "rows" of the graphical screen are placed. The two "Separate Rows" options result in a "Windows like" look, but use more vertical screen space.

Option	**Description**
Adjacent Rows	Makes entry fields on adjacent rows overlap and share a common border (if boxed). This option uses the minimum screen height.
Separate Rows	Places a small amount of space between the adjacent rows such that the entry field boxes are clearly separate from each other.
Separate Rows / 3D	Gives the same result as the "Separate Rows" option, except that each entry field has a 3-D border drawn for it.
Font	Lets you pick the font that is used for all the controls (individual controls can be changed in the Screen Designer). Note that many fonts designed for word processing are taller than the built-in fonts for a particular point size. These fonts have additional white space built into them on the top and bottom edges and therefore use more vertical screen space than the built-in fonts.
Small	Displays your control with the runtime's built-in proportional small font.
Medium	Displays your control with the runtime's built-in proportional medium-sized font.
Large	Displays your control with the runtime's built-in proportional large font.
Fixed	Displays your control with the runtime's built-in fixed-width font.
Traditional	Displays your control with the runtime's built-in traditional font, which uses a different underlying character set than Windows normally does. The traditional font mimics IBM's original character set.
Custom	Allows you to select any font on the system for your controls. This option is available once you click on the Pick button.
Pick	Displays the Font dialog box, in which you can select the font, its style (e.g., italic), and size for your control.
Use Alternate Rules	Causes the wizard to re-evaluate the original character screen using slightly different conversion heuristics. Generally speaking, the alternate rules work well if you are getting too many entry fields using the default rules.

Option	**Description**
Size	Displays the current screen dimensions in pixels. This is useful when you want to ensure that the resulting window fits on a specific resolution screen. For example, you may want to make sure that the resulting screen is not larger than 800 x 600 pixels.
OK	Finishes the conversion process. The wizard imports the graphical screen to the AcuBench Screen Designer and returns control to the original character program.
Cancel	Cancels the conversion process. The converted screen is destroyed along with the editing window and control returns to the original character program.
Set Default	Records the current Properties dialog box settings in the registry. These settings become the default values for future screen conversions.

Any changes made in the Properties dialog box are immediately reflected in the graphical window.

Three changes can be made in the graphical window itself:

1. You can change the size of the window by grabbing one of the edges with the mouse and dragging to the desired size. The “Size” portion of the Properties dialog tells you the current size of the window in pixels.
2. The wizard uses a variety of heuristics to decide whether a field in the character application should be represented as a label or an entry field. Sometimes, these do not produce the desired result. You can change the type of the control by right-clicking on the control. Select the desired control type from the pop-up menu.
3. If desired, you can delete a control from the screen produced by the wizard. You would normally remove controls that don’t make any sense. For example, you might want to delete a menu bar that has been turned into a set of controls. Later you can reconstruct the menu bar in the Screen Designer. To delete a control, right-click on it and select **Delete**.

When you are finished editing screen properties, click **OK**, terminate your program normally, and control returns to AcuBench. Use the AcuBench Screen Designer to further manipulate your screens as required.

24.5.8 Manipulating the Screen in the Screen Designer

After converting your screens with the Character-to-GUI Wizard, you can use the AcuBench Screen Designer and all of its features to modify the screens to your liking. You can add graphical controls, define control properties, modify grid behavior, and perform any other function that you might if you were designing the screens from scratch in AcuBench. You can make changes all at once, or over time, as needed. For more information about modifying your screens, see **Chapter 14, "Chapter 14: Working with Screens."**

You may want to move your newly imported screens into their host programs in the workspace Structure view. You can do this by clicking on the new screen in the workspace and dragging it into the screen node in the desired program.

Be sure to review the characteristics of each screen that you converted. As you review, you can make minor modifications to the look and feel of the screen and resize the screen as necessary. (Be aware that in Windows, the size of an initial screen is different than it might be in a character-based application.) Use the alignment functions to position and size controls, lock controls, and verify the tab order. In addition, you may want to change the names of your screens or rename individual controls in the Property window to give them meaningful names. If your original screens were constructed from a Screen section, we suggest that you name the screens with the same name that you used previously.

In many cases, you should replace consecutive singular entry fields with multi-line entry fields. You should also check the window type of your screens. Remember that there can be only one "standard" window. Subsequent windows are either floating windows or associated with pre-existing window handles. After you've performed these functions, you should verify that font, title, value, and other variables are set correctly.

Finally, if you want your menu to be associated with the screen (and have code generated to show it), be sure to update Main Menu in your screen's Property window.

Any other changes that you want to make to your screens can be made now if you like. How far you take your screens is up to you.

The following list offers some guidelines for modifying your newly converted screen:

1. Examine the window properties of the imported screen. Set the following properties:

   Title: If the window in question is to be displayed on another Window's handle, the title is just documentation. It is useful when working on a screen to have a quick visual cue of the name of the screen when it is not a primary screen.

   Screen Name: If the original screen came from a Screen section, preserve the screen's name to facilitate integration of code.

   Window Type: This always defaults to "Standard". However, there can be only one "standard" window in a program. If the window in question is to be displayed on another window's handle, this is not an issue. If not, select another window type.

   Size / Lines: We suggest that you conform to some basic window size/lines measurements, and set all windows to those measurements.

2. Examine the control properties of the imported screen.

   a. Set Control Name.

   b. Verify that Value-Variable and Title-Variable have been correctly preserved by the import process.

   c. Verify that embedded procedures have been correctly preserved by the import process.

   d. Align the controls. Conform alignments to preserve application "look and feel". Conform the line number on which status messages appear (typically, line 24 in a character-based application).

3. Examine the screen's ACCEPT statement. ACCEPT Screen-Name does not require modification. ACCEPT Variable-Name does, because variable-name is now part of a screen. Refer to **section 24.5.10** for more information.

4. Right-click on the screen and select the **Change Prefix** command. This action is in preparation for moving the screen to the host program, where all screen prefixes must be unique. Select **Change Prefix Only**. Assign a unique prefix to the screen.

5. Drag-and-drop the screen in the host program. Delete the program structure file when it contains no more screens.

6. Examine the screen's window-handle. If the screen is to be displayed on another screen, then change the window handle to the handle of that screen. If the screen is a floating window (derived from a pop-up window), make sure it shares the same handle as the original pop-up window.

7. Save.

24.5.9 Generating New Screen Section Code

When you are done modifying your new screens, use the **Build/Generate** command to generate a COBOL code source file (".cbl") for the screen. As with all AcuBench screens, the code generation process also gives you a Screen section file (".scr"), an event paragraph file (".evt"), a menu paragraph file (".mnu"), a Procedure Division file (".prd"), and a Working-Storage file (".wrk").

Note: Before you generate code for your screens, you should be familiar with the code generation model used by AcuBench. See **section 3.3** for details.

24.5.10 Integrating Code Back into Your Program

The final step in the conversion process is integrating your new graphical screens back into your program. This step typically takes the most time and may require some program changes. In some cases, the changes can be extensive. The amount of work you have to do depends on the original state of your program. The process is made somewhat easier because the new screen refers to your program's variables and embedded procedures where applicable.

There is no single correct way to do this step, and no procedure that works for every case. Instead, you have to evaluate your program to see how it can best interact with the updated screen. In the simplest case, the integration process consists of putting the new code in place and commenting out the old code.

When you're integrating the new code, you should analyze your application's:

- data characteristics
- keyboard characteristics
- display characteristics
- initialization sequence

Data characteristics

Even though the concept of the Character-to-GUI Wizard is display oriented, you must understand the relationship between the screen and the data. For instance, you should verify that the fields on your new screen are matched with the fields in the FD or working-storage. You should also verify that embedded procedures are associated with your screens and its fields correctly.

In addition, if you've done things like create data sets, then you have to be aware of the fact that you generated code that opens files on initialization. If you don't need those files open, you should de-select OPEN in the definition of the data set. To do this, right-click on the data set node for the target program in the workspace Structure view and select **Referenced FD/SL files** from the drop-down box. Clear the **Open** check box for the files you do not wish to be opened.

Keyboard characteristics

When integrating your new screen code, you must also be aware of your program's keyboard characteristics. If your application instructs users to press a function key and you want to implement this functionality with a push button, you have to know what exception value that key produces and you have to match the same exception values to your screen's new push buttons.

If you printed your "cblconfig" file or the section of code that contains environment settings before conversion, you should be able to easily locate the KEYSTROKE settings.

Display characteristics

The most important element of the conversion is your application's display characteristics. In general, the conversion process results in a set of new DISPLAY statements that you must use in place of your old DISPLAY statements.

To locate your old DISPLAY statements, you can select **Find All** in the Find dialog box and search on the word "DISPLAY". This generates an Output window list of all the lines of code in your program containing the word, "DISPLAY." If it is not obvious to which screen the DISPLAY statements relate, try running the ACUCOBOL-GT debugger at the same time that you capture the screens with the Character-to-GUI Wizard. If you run your program with both the "--char2gui" and "-d" runtime options, you can turn on paragraph trace, capture a screen, turn off paragraph trace, and know exactly where in the source you've been.

Once you know which DISPLAY statements correspond to which screen, you can begin replacing old code with new.

Where screens are displayed initially, you should comment out the old DISPLAY statement and replace it with a PERFORM statement that uses the following syntax:

```
PERFORM Acu-[screen-name]-Scrn
```

For example:

```
    * display myscreen
     PERFORM Acu-myscreen-Scrn
```

This paragraph contains code to create a window, display the screen and its main menu on the window, and initialize any complex controls that have been included.

Where elements of screens are displayed (e.g., with field-level DISPLAY statements), you should comment out the old DISPLAY statement and replace it with a MODIFY statement. For example:

```
    * display id-no-field
     MODIFY ID-NO-FIELD value Techhelp-ID
```

This is made easier because the wizard preserves the name of the Screen section element.

If your original program uses Screen section DISPLAYs and you have duplicated the name of the screen in the Screen Designer, then you may have very little work to do. However, you should replace the initial "display window ERASE" statement with the workbench-generated DISPLAY STANDARD WINDOW code in the ".prd" file.

Be sure to comment out non-Screen section DISPLAY statements if your program uses a combination of Screen section and field-level DISPLAY statements. DISPLAY LINE and DISPLAY BOX statements fall into this category. Comment them out.

ACCEPT handling

Even though the generated code contains ACCEPT statements as well as DISPLAY statements, we recommend that you leave your application's ACCEPTs intact whenever possible. Where your application ACCEPTs screens, no conversion is necessary. (The screen name is already established by the wizard.)

However, where your application ACCEPTs variables, you should comment out the old syntax and replace it with an ACCEPT screen. You should then add an entry field to the screen whose value is the variable you were previously ACCEPTing. For example:

```
    * accept new-status
     ACCEPT NEW-STATUS-SCREEN
```

In this example, the screen, NEW-STATUS-SCREEN, contains an entry field whose value is new-status. The old ACCEPT is commented out and the new ACCEPT screen is added.

As a rule, you should examine your ACCEPT statements to determine whether they are ACCEPTing screens or variables, and therefore, whether or not you must perform any conversion. To locate the ACCEPT statements, you can select **Find All** in the Find dialog box and search on the word "ACCEPT".

Other display considerations

After you have made an initial pass through your application's ACCEPT and DISPLAY statements, you should consider how and whether your screens may be used differently in a graphical world than they were in a character-based world. For instance, it is a common practice in many character-based applications to use DISPLAY statements to redisplay a main screen whenever a field is updated. In the graphical world, it is impractical and unnecessary to recreate a window and redisplay an entire screen just to update a field. Therefore, you may want to use a MODIFY statement to update the field rather than the PERFORM screen.

In addition, in character-based applications, DISPLAYs are often made over existing DISPLAYs, in effect replacing the old DISPLAY with the new one. In graphical applications, controls must be explicitly destroyed or made invisible before other controls are put in the same space.

The way that a screen closes in a character application is also different from the way a screen or window is closed in a graphical application. For this reason, when you are replacing your DISPLAY statements, you should track where the screen closes or is blanked (a less obvious display event) and convert the close as needed. For example, you may replace a DISPLAY statement with a MODIFY statement in the following case:

```
    * display grow-scrn-1.
     MODIFY GROW-SCRN-1 TITLE GROW-DESC-LABEL.
```

However, if you are using a pop-up window, then your close event may specify “close window handle-1”. In this case, when you import the screen, all you need to do is give it a handle of “handle-1” and no other conversion is necessary.

Another important difference between character and graphical applications is that character-based applications often dynamically build a screen one line at a time, but it is impractical to display elements line-by-line in the graphical world. Therefore, if your application DISPLAYs a variable in column 10, line 10, and then again in column 10, line 11, and so on, in the graphical world, this may be better accomplished through the use of a list box.

Finally, with a graphical application, you have the opportunity to display message boxes, combo boxes, and other graphical elements that you didn’t have access to before. You can use the Screen Designer to add these elements whenever you’re ready.

Initialization sequence

When you generate code for a screen, a statement called “perform acu-initial-routine” is generated. The acu-initial-routine paragraph performs several functions, some of which you may want and some of which you may not. It performs an “ACCEPT System-Information FROM System-Info” and an “ACCEPT Terminal-Abilities FROM Terminal-Info”. In addition, it initializes fonts, bitmaps, and ActiveX resource files; creates menus; and opens files described in data sets. Your character application may already be performing some of these functions, but it isn’t likely to perform all of them, and some, like initializing fonts, are critical in the graphical world.

As you are adjusting your original application for the new screen code, you need to make some decisions about what you want your application to do upon initialization. If you want all of the stated initialization routines to take place, you can leave the new perform statement in place, but you may need to remove old redundant code. If you want to perform some of the initialization routines but not others, rather than performing acu-initial-routine, you may prefer to call individual routines, like initialize fonts. In this case, delete the new perform statement and add the desired calls to your own sequence.

24.5.11 Tips and Techniques

To simplify your conversion process, consider performing the following steps before converting your text-based screens:

- Identify all character-based DISPLAY code in your application.

Note: Character-based DISPLAY events include DISPLAY statements and CLOSE WINDOW statements.

- If your screens are defined in a Screen section, print descriptions of your screens. These can be used to ensure consistency, particularly regarding screen name, control names, and variable names associated with controls.
- Create a screen flow diagram, that is, a list of the screens you are converting along with instructions on how to bring up those screens and conditions that need to be present in order to bring them up. This helps you determine whether you can capture all of the screen conditions contained in your application, or whether you must run the application again with another set of conditions to capture more screens. It also helps you associate the new screen forms with their host programs when you are ready to integrate the code back into your application.
- Print descriptions of your menu bars. Menu bars are not captured by the wizard and must be rebuilt "manually" in the Screen Designer.
- Print SELECT statements and file descriptors (FDs).
- Print your "cblconfig" file or the section of code that contains environment settings. KEYSTROKE settings are important if you want to replace function key (F-key) labels with push buttons.
- Back up your original application. The copy from which you are converting quickly becomes a "work in progress," and you need to have the original system to verify that functionality has remained intact, as well as to serve as a reference point.
- Create an AcuBench project and copy the necessary source files, data file selects and descriptions, and other COPY files into the created folders. Add source files to the project using the Add/Remove Files

function and parsing for COPY files. This creates a program structure file in the workspace Structure view. Note that only programs with screens that will be converted need to have program structure files created for them.

- If the application that you plan to convert has calls to ACUCOBOL-GT's "menubar.cbl" program, replace these calls with calls to "menubar2.cbl" before capturing the screens. The "menubar.cbl" program creates a character-based menu bar and "menubar2.cbl" creates a graphical menu bar using the same input.
- If you think you may have trouble creating "real" conditions for a screen to appear, run your program with the debug option "-d". In debug mode, you can set breakpoints where the program evaluates the state of certain fields and you can set them using the A (accept) function. In this manner, you can force the display of the screen you want to capture.

In addition, be careful not to distribute the wizard to your end users when you ship the runtime. Specifically, do not give them the file "achr2gui.dll". This file is normally located in the same directory as the runtime (which defaults to the "bin" subdirectory where you installed ACUCOBOL-GT). This file is not needed except when running the wizard.

24.6 Importing a Graphical Screen

This section describes the AcuBench graphical Screen Import Utility, which translates graphical ACUCOBOL-GT screens into those that can be read in the Screen Designer. The Screen Import Utility creates a text file that "defines" a graphical screen created by an ACUCOBOL-GT program. This utility is used to import graphical screens into AcuBench, and uses the ACUCOBOL-GT runtime to produce a description of the screen. For information about importing character screens, refer to **section 24.5** in this manual.

The following steps summarize the use of the Screen Import Utility. The remainder of this section describes aspects of the import function in more detail.

1. In the Project Settings dialog box, set the compiler to use the “-Zw” option.
2. Also in the Project Settings dialog box, set the runtime to use the “-import” option.
3. Compile and execute the screen(s) to be imported using the above options.
4. Right-click in the screen to be imported and select the **Import Screen** command.
5. Close the program that contains the screen(s) you just imported. The imported screen(s) should be represented in your workspace in a new program. At this point, the imported screen can be edited, the screen can be saved as an “.stf” file, and code can be generated for the imported screen. You can also move the imported screen(s) to the program of your choice in the workspace.

24.6.1 Setting Compiler Options

For best results, the program containing the screen to be imported should first be compiled with the ACUCOBOL-GT compiler using the “-Zw” option. This option, which is set through the AcuBench Project Settings dialog box or on the ACUCOBOL-GT command line, causes the compiler to insert information that may be lost during compilation. Such information includes, for example, the names of the variables used to create the screen items. Importing screens without using the “-Zw” option yields screens that are not as complete as they are when you use the “-Zw” option.

When compiling, you can also select a mapping option to define how your data names appear in the screen description: in uppercase, lowercase, or mixed case. By default, uppercase data names are displayed. Note that the case transformation occurs only for characters in the ASCII character set (the English alphabet). For characters outside of the ASCII character set, no transformation is performed.

You can set the “-Zw” compiler option using the following steps:

1. Use the **Project/Settings** command to display the Project Settings dialog box. Choose the **Compiler** tab.

2. Select **Miscellaneous Options** in the catalog drop-down box. Check the “-Zw” option, “Prepare for use with the Screen Import Utility.”
3. Select **OK** to save the setting.
4. Select **Save as Default** to save the setting in the “default.pof” file. These settings are applied to new projects when they are created.

Set mapping options as follows:

1. Use the **Project/Settings** command to display the Project Settings dialog box. Select the **Compiler** tab.
2. Select **Mapping Options** in the catalog drop-down box.
3. Select the option that best fits your needs.

-Mu	Names are uppercase
-Ml	Names are lowercase
-Mm	Names are mixed case: upper for the first letter of each word, lower for the rest

4. Select **OK** to save the setting.
5. Select **Save as Default** to save the setting in the “default.pof” file. These settings are applied to new projects when they are created.

24.6.2 Setting the “-import” Runtime Option

After the relevant programs are compiled, you need to execute them. Perform this step as if you were running the programs normally, except that you should add the “-import” option to the runtime command line. The ability to add the “-import” option is managed through the Project Settings dialog box as follows:

1. Select the **Project/Settings** command to display the Project Settings dialog box. Choose the **Runtime** tab.
2. Select **Common Options** in the catalog drop-down box. Check the “-import” option, “Import screen file.”

3. Select **OK** to save the setting.

4. Select **Save as Default** to save the setting in the “default.pof” file. These settings are applied to new projects when they are created.

Alternatively, to select the “-import” option, you can right-click the program node in the File tab and select the **Screen Import** command. Note that only the Windows 32-bit runtime supports the “-import” option. Also, you must have the file “WEXPRT32.DLL” installed in the same directory as the runtime. This “.DLL” comes with the 32-bit Windows runtime, but is used only for the import function.

24.6.3 Graphical Screen Importing

After you have compiled and executed the program, you are ready to import. The compile and execute functions are available as separate commands in the menu bar or the Project toolbar. You can also right-click on the source file in the workspace File view and select the **Screen Import** command.

Perform the screen import operation as follows:

1. With the relevant program running, right-click in your screen to display the Import Screen pop-up menu. Make sure you click the mouse in the screen itself and not over a control on the screen.

2. To “capture” the screen for import, select the **Import Screen** pop-up menu with the left mouse button.

   You may import multiple screens in a single execution. The results are all saved together in a single text file named “import.out”.

3. Close the program that contains the screen(s) you just imported.

   The imported screen should now be represented in the workspace Structure view in a new program, from which it can be opened, modified as needed, saved, and have code generated for it. Multiple imported screens are opened “cascade” style in the Screen Designer.

4. If you want to add the new “import.out” file to another program in the workspace Structure view, right-click on the program node in the workspace, and select the **Add Screen** command from the pop-up

menu. Select "**.out**" in the "Files of type" drop-down box. If necessary, use the **Browse** button to navigate to the desired "import.out" file. You can also move the imported screen to the program of your choice by clicking on the imported screen in the workspace and dragging it to the desired program screen node.

24.6.4 Graphical Screen Importing Notes and Restrictions

- The grid, tree view, Web browser, status bar, and ActiveX controls, as well as pop-up menus, are only partially imported. For these controls, common properties and styles are imported. However, control-specific properties and styles are ignored and must be added manually.
- Only controls are imported. Screen elements described by textual fields (e.g., DISPLAY "CUST NO") are not imported.
- The imported screen does not retain any of the structural information in the original source's Screen section. For example, if a Screen section item contains subsidiary group items, that information is lost. The Screen Designer cannot represent these items, so they are not imported. The resulting screen has the same look as the original, but a different internal structure.
- Importing a screen and then generating code may result in an illegal COBOL program. This can happen when the original source code contains structures that have no representation in the Screen Designer. For example, if the original program contains two fields that have the same name, but are distinct using qualification by intermediate group items, the resulting program may be illegal as the intermediate group items are lost. In another example, if the USING phrase for a control names a qualified data item (e.g., MY-DATA OF MY-GROUP), then the resulting program contains an illegal data declaration (e.g., "01 MY-DATA OF MY-GROUP PIC X(10)"). You can adjust for these anomalies in the Screen Designer after importing the screen.
- If both FROM and TO are specified, the TO item becomes the control's VALUE and the FROM item is ignored. Otherwise any specified FROM/TO/USING item becomes the control's VALUE. The control's actual VALUE is suppressed, but the VALUE data item name and picture are imported.

- CLINE, CCOL, CSIZE, and CLINES are not currently imported.
- MULTIPLE VALUE items have the word “Table” in front of their data item name. However, the number of occurrences in the table is not imported.
- For entry fields, list boxes, and combo boxes, if neither BOXED nor NO-BOX is specified, the import utility sets BOXED to “true” (although FIELDS-UNBOXED can affect this). While this is an accurate reflection of the control's appearance under Windows, this could change the look under character systems. You might want to leave both BOXED and UNBOXED “false” and have the Screen Designer assume the BOXED appearance. Also note that FIELDS-UNBOXED implies the NO-BOX style (for entry fields) at the time the control is created. This is reflected in the imported code even though NO-BOX is not specified in the original source.
- Labels always have a LABEL-OFFSET of “0”. The label offset value is reflected in the coordinates for the label.
- Single-line entry fields report their MAX-LINES as “0”, because MAX-LINES is only meaningful for multi-line entry fields.
- CURSOR and ACTION properties are not imported for entry fields. These fields are not normally used in a Screen section.
- For list boxes, the following properties are not imported: MASS-UPDATE, ITEM-TO-ADD, RESET-LIST, ITEM-TO-DELETE, INSERTION-INDEX, SEARCH-TEXT, SELECTION-INDEX, QUERY-INDEX, and ITEM-VALUE. All of these are dynamic properties that are not part of the static appearance of the list box. You may want to reconsider ITEM-TO-ADD and possibly others.
- For combo boxes, the following properties are not imported: MASS-UPDATE, ITEM-TO-ADD, RESET-LIST, ITEM-TO-DELETE, and INSERTION-INDEX.
- For frames, FILL-PERCENT is not imported.
- For tab controls, RESET-TABS and TAB-TO-DELETE are not imported.

- For bars, one dimension has a size of zero. Use the value of the WIDTH property to determine the actual width of the line.
- For bitmaps, either SIZE or LINES may be zero. In this case, the full size of the actual bitmap should be used for the corresponding dimension.

Index

A

About AcuBench command 22-2
ACCEPT-CONTROL variable, setting 4-22
ActiveX Control Components command 15-21
ActiveX Control Properties dialog 15-22
ActiveX controls 15-20
- adding 15-22
- licensing issues 15-22
- removing 15-23
- setting the initial state 15-22

ActiveX, Special Names COPY file 4-23
activex-def tags 4-23
acu4glfd utility 19-21
AcuBench file types 3-10
AcuBench toolbars
- adding 4-31
- adding buttons to 4-32
- customizing 4-31
- deleting 4-31
- removing buttons from 4-32
- spacers in 4-32

AcuBenchPrint.dll 17-29, 17-32
ACUCOBOL-GT
- character screen import 24-11
- definition COPY files 4-24
- documentation set 2-4
- importing graphical screens into the Screen Designer 24-30
- utilities 19-2

AcuConnect 1-5
- printing reports in thin client 17-32

Acucorp technical support 1-6
acu-def tags 4-24

ACUPATH environment variable 7-13
Acu-Report1-ThinPreview 17-35
acuthin --wait 9-28
AcuXML 19-22
AcuXUI 14-42
Add FD/SL command 8-29
Add Keyword Set dialog 11-9
Add New Template File dialog 12-17, 13-6, 16-7
Add Paragraph dialog 14-37
Add Screen command 14-40
Add Screen to Program dialog 14-40
Add to Watch Window command 20-5
Add/Remove Files command 6-13, 22-4
Add/Remove Project Mode dialog 7-4
adding a child item to a tree view 15-18
adding a keyword 11-9
adding a menu item 15-24
adding a project mode 7-4
adding a screen template 13-6, 16-7
adding a submenu item 15-25
adding a tree view item 15-18
adding ActiveX controls 15-22
adding configuration variables 7-9
adding controls 14-9
adding files 6-14, 9-14
adding tags 4-26
adding user-defined variables 7-12
AfterDoPrint event 17-19
    code samples 17-21
    described 17-21
    generated files 17-24
    modifying source file 17-24
Align commands 14-30
Align Control commands 14-31
Align menu 14-30

Align toolbar 22-8
aligning controls 14-30
ANSI to Terminal Format command 12-4
application development approaches 3-12
areas A and B 11-6
augment function, vutil 19-16
Auto Step command 20-12, 22-7
auto-alignment option 13-3, 16-3
auto-increment control variable values 14-33
auto-indent function 4-15, 11-8
Autoload controls 14-14
automated code generation 3-7
automatic save 4-6
AutoStep Properties dialog 20-12
AXDEFGEN 15-21

B

Background list box 11-6
bar control 15-5
Before and After Print event 17-18
Before and Ater events
    described 17-20
BeforeDoPrint event 17-19
    code samples 17-21
    described 17-21
    generated files 17-24
    modifying source file 17-24
bitmap control 15-5
bookmarks 12-7
Boomerang 9-24
breakpoints
    Breakpoints command 20-14
    modifying 20-17

  setting and removing 20-15
  using bookmarks as 12-7
  viewing 20-17
Build Environment options 4-9
Build Graphical Screen command 24-17
Build menu
  Build Workspace 6-4, 22-4
  Generate 14-40
  Rebuild Workspace 6-5
  Regenerate Workspace 6-6
  Stop Build 6-6
build scripts 3-23
Build Workspace command 6-4, 9-24
building from the command line 3-22

C

C$COPY library routine 17-34
Call Stack command 20-7
Call Stack window 22-6
Capitalize command 12-8
cblutil 19-3
  -info 19-6
  -lib 19-4
  -native 19-8
Cell Setting tab 18-21, 18-23, 18-25
Center Horizontal and Vertical commands 14-31
CFE (Configuration File Editor) 7-9
Change Prefix dialog 14-4
changing a screen's unique prefix 14-4
changing column width in source 11-6
changing menu bars 15-27
changing project directories 3-14
changing toolbars 15-30

changing variable values of multiple, identical controls 14-33
--char2gui runtime option 24-16
character screen conversion 24-14
character screen import 24-11
  benefits and restrictions 24-12
  code integration 24-23
  implementation tips 24-28
  manipulating the screen 24-21
  Properties dialog 24-17
Character Screen Import command 24-17
character-based screen controls 14-10
character-based screens 24-11
Character-to-GUI Wizard
  benefits and restrictions 24-12
  --char2gui runtime option 24-16
  code generation 24-23
  code integration 24-23
  implementation tips 24-28
  importing character-based screens 24-11
  mapping options 24-15
  Properties dialog 24-17
  using 24-14
  -Zw compiler option 24-15
check box control 15-6
child item, in tree view 15-18
CLASSPATHDIR environment variable 14-42
Clear All Bookmarks command 12-7
Clear All Breakpoints command 22-6
Close Workspace command 6-6
closing a workspace 6-6
code completion function 4-16, 12-13
Code Editor
  keyboard shortcuts 23-24
  line length 11-5
  opening a file 12-2

- text block handling 12-11
- trailing spaces in 11-5

Code Editor options
- and ANSI display fields 11-6
- auto-indent values 4-15, 11-8
- Background list box 11-6
- Code Insight options 4-16
- color functions 11-6
- column width 11-6
- converting tabs to spaces 11-8
- disabling color functions 11-6
- display in Lists 4-19
- display tab function 11-8
- end-of-line marker 11-4
- line length 11-4
- line number pane width 11-4
- modify code 11-5
- Sequence Number area 11-6
- setting tabs 11-8
- sorting lists 4-17
- source format 11-5, 11-8
- tab size function 11-8
- tabs support 11-8
- text type 11-6
- virtual space 11-4

Code Editor options, listed
- Constant List 4-17
- COPY File List 4-17
- Font 11-7
- Format 4-15
- General 4-15, 11-4
- Keyword 4-15, 11-8
- Paragraph List 4-17
- Tabs 4-15
- Variable List 4-17

Code Editor windows, splitter 12-8
code generation 3-7
  ACUCOBOL-GT definition COPY files 4-24
  COPY statements 3-7
  DECIMAL POINT IS COMMA 4-24
  development approaches 3-12
  Event COPY files 4-26
  event procedure COPY file 3-8
  external COPY files 4-25
  file creation 3-16
  file descriptor COPY files 4-24
  file handling 8-22, 10-6
  Linkage section COPY file 3-9, 4-25
  menu COPY file 3-8
  Procedure COPY files 4-26
  Procedure Division COPY file 3-9
  program comments 4-23
  program file 3-9
  program identification code 4-23
  reports 17-23
  screen COPY files 3-8, 4-25
  Screen section 24-23
  SELECT statement COPY files 4-24
  Special Names for ActiveX 4-23
  tags 4-22
  Working-Storage COPY files 3-9, 4-25
Code Generator options
  Generate Document 4-20
  generated file attributes 4-21
Code Insight functions 4-16, 12-12
code integration
  data characteristics 24-24
  display characteristics 24-25
  initialization sequence 24-28
  keyboard characteristics 24-24

code parameters 4-16
code template
  adding to project 12-17
  function 4-17
Code Template command 12-17
CODE-PREFIX environment variable 7-13
color functions 11-6
Column Setting tab 18-23
combo box control 15-6
command-line interface 3-22
  rebuilding the workspace 6-5
  regenerating the workspace 6-6
comment tags 4-22
common properties 15-2
common screen options 14-29
Compile command 22-4
compile options
  debugger 20-8
  extra symbol debugging 20-8
  mapping 24-15, 24-31
  minimal symbol debugging 20-8
  setting 7-7
  source-level debugging 20-8
COMPILERNAME environment variable 7-13
compiling 9-21
  from the AcuBench command line 3-23
  multiple programs 9-24
  to a remote server 9-22
  Use Thin Client 9-22
concurrent project development 3-2, 6-7
Configuration File Editor (CFE) 7-9
  adding a variable value 7-10
  adding/editing user-defined variables 7-12
  locating variables 7-11
  removing a value or comment 7-11

searching 7-11
setting or changing a comment on a variable 7-11
setting or changing the value of a variable 7-10
Constant List box 12-15
Constant List Code Editor options 4-17
context-sensitive help 2-3
controls 14-29, 15-2
ActiveX 15-20
adding 14-9
copying 14-45
drawing 14-10
properties 15-2
standard 15-3
Copy command 14-33, 22-2
COPY File List box 12-16
COPY File List Code Editor options 4-17
COPY files 3-7
adding to a project 9-16
COPY statements, using in code generation 3-7
Copy to Clipboard command 12-16
copying controls 14-33
copying screens 14-45
COPYPATH environment variable 7-13, 7-14
modifying 9-16
copy-procedure tags 4-26
copy-screen tags 4-25
copy-working tags 4-25
Create Alias command 22-5
Create Alias dialog 9-27
creating a file alias, thin client 9-27
creating a keyword set 11-9
creating a menu bar 15-24
creating a new data layout file 8-5
creating a new program 9-3
creating a pop-up menu 15-27

creating a project 6-7
creating a screen 14-2
creating a toolbar 15-29
creating a tree view 15-17
creating a workspace 6-3
creating an object library 19-4
creating reports 17-2
  detail section 17-6
  Drag-and-Drop command 17-14
  group headers and footers 17-6
  page headers and footers 17-6
  report box 18-6
  report check box 18-7
  report control properties 17-16
  report date time 18-8
  report entry field 18-10
  report grid 18-12
  report headers and footers 17-6
  report image 18-14
  report label 18-16
  report line 18-16
  report occurs 18-17
  report radio button 18-19
  report table 18-20
  section properties 17-11
creating screen templates 14-40
creating XFD files 8-25
customizing a template 4-8
customizing AcuBench toolbars 4-31
customizing the user interface 4-2
  Environment options 4-5
  loading options 4-4
  saving options 4-4
Cut command 12-5, 14-33, 22-2
cutting a control 14-33

D

data definition, project level 8-2
data design tools 8-2
Data Designer options 4-28
  file descriptor options 4-29
data File Designer 8-10
data layout file
  adding to a project 8-29
  creating from existing COPY files 8-7
  described 8-3
  properties 8-31
  refreshing 8-31
  reusing 8-29
Data Set Designer 10-2
Data Set Member Files dialog 10-10
data sets 10-2
  upgrading from Version 5.x to Version 6.x or later 4-22
Data View 3-5
date entry control 15-7
Debug (Runtime) command 22-4
Debug Environment options 4-10
Debug menu 20-9
  Go 20-8
  Interrupt 20-9
Debug Mode 20-7
Debug toolbar 20-4, 22-6
debugger 20-2
  Add to Watch Window command 20-5
  Auto Step command 20-12
  Breakpoints command 20-14
  Breakpoints dialog 20-15
  Call Stack command 20-7
  compile options 20-7
  entering 20-8

Exit Debugger command 20-12
Go command 20-8, 20-12
highlighting 20-3
Interrupt command 20-9, 20-12
Memory command 20-7
Memory window 20-7
modes 20-7
modifying breakpoints 20-17
monitoring variables 20-18
output file 20-4
Output window 20-2
Quick Watch window 20-6
Quit Debugging command 20-12
Record Script command 20-14
Restart command 20-12
Run Script command 20-14
Run to Cursor command 20-14
runtime 20-2
setting a monitor 20-5
setting and removing breakpoints 20-15
Skip to Line command 20-13
starting a program in 9-27
Step Into command 20-13
Step Out Paragraph command 20-13
Step Out Program command 20-13
Step Over command 20-13
switching threads 20-19
Threads command 20-19
Trace File command 20-10
Trace Flush command 20-10
Trace General command 20-10
Trace Levels command 20-11
Trace Option command 20-10
Trace Paragraphs command 20-10
Trace Screens command 20-10

Trace to Debug Window command 20-11
using with the thin client 9-28
viewing breakpoints 20-17
Watch command 20-5
Watch window 20-4, 20-5
windows 20-3
working directories 20-2
Workspace window 20-2

debugger commands 20-10
debugger Stack Info window 20-6
debugging a program 9-27
DECIMAL POINT IS COMMA 4-24
decimal-point tags 4-24
declarative tags 4-25
declaratives, IO Handling tab 8-22
Default Report Writer options 16-5
Default Screen Designer options 13-4
defining a configuration variable 7-10
deleting a keyword 11-9
deleting a keyword set 11-9
deleting a menu item 15-25
deleting a tree view item 15-19
designing a toolbar 15-29
detail section 17-6
development approaches 3-12
- general considerations 3-13
- new applications 3-17
- traditional 3-19

development window 2-12
dialog box
- ActiveX Control Components 15-21, 15-23
- ActiveX Control Properties 15-22
- Add New Template File 13-6, 16-7
- Add Paragraph 14-37
- Add Screen to Program 14-40

- Add/Remove Files 6-13
- Add/Remove Project Mode 7-4
- Breakpoint Details 20-16
- Breakpoints 20-15
- Change Prefix 14-4
- Data Set Member Files 10-10
- File/New 12-3
- Find 21-2
- Find in Files 21-4
- Font 11-7, 24-19
- Layout Data Settings 14-28
- Menu Designer 15-24
- New FD/SL 8-5
- New Folder 6-12
- New Screen 14-3
- Page Setup 2-15
- Print 2-14
- Print Setup 2-14
- Project Settings 24-16, 24-31
- Properties 24-17, 24-18
- Replace 21-9
- Replace in Files 21-11
- Reset Controls Tab Order 14-34
- Table Setting 18-23
- Threads 20-19
- Trace Level 20-11

Disable Breakpoint command 22-6
display in Constant List 4-19
display in COPY File List 4-19
display in Paragraph List 4-19
display in Variable List 4-19
display property in Property window 13-5, 16-6
display tab function 11-8
displaying object file information 7-18, 9-22
displaying object library information 19-6

DLT
  defined 8-3
  properties 8-31
document notation 2-5
documentation overview 2-2
Drag and Drop
  autoload 14-14
  command, Report Composer 17-14
drawing controls 14-10

E

Edit menu
  Bookmark 12-7
  Clear All Bookmarks 12-7
  Code Template 12-17
  Find 21-2
  Find in Files 21-4
  Find Scope 12-13
  Go To 12-6
  Go to Next Error 12-17
  Go to Previous Error 12-17
  Insert File 12-9
  List Constants 12-14
  List COPY Files 12-16
  List Paragraphs 12-14
  List Variables 12-14
  Next Bookmark 12-7
  Open COPY File 12-16
  Previous Bookmark 12-7
  Verb Block Match 12-13
editing runtime configuration files 7-9
editing user-defined variables 7-12
Editor toolbar 22-6

embedded procedure COPY file 3-8
end-of-line marker 11-4
entry field control 15-8
entry mode for a grid control 15-10
entry-befprg tags 4-26
Environment options 4-5
    automatic save 4-6
    automatically create a program file 4-14
    Build 4-9
    COPYPATH 4-14
    customize template 4-8
    Debug 4-10
    default source format 4-6
    extended ASCI characters 4-7
    force variable check on load 4-6
    General 4-5
    Keyboard 4-11
    Miscellaneous 4-13
    on add project 4-6
    on open workspace 4-6
    on start-up 4-6
    Prefix 4-12
    recently used list 4-6
    Template 4-7
    Use File/New 4-7
    Version Control 4-8
    when building the workspace 4-10
    when compiling programs 4-9
    when executing source files 4-9
    when running tools 4-9
environment variables 7-12
    ACUPATH 7-13
    adding 7-14
    changing 7-14
    CLASSPATHDIR 14-42

- CODE_PREFIX 7-13
- COMPILERNAME 7-13
- COPYPATH 7-13, 7-14
- deleting 7-14
- RUNNAME 7-13
- XUIJAR 14-42

error file 20-4

Event COPY files 4-26

Event Editor 14-35
- accessing from Data view 8-30
- Add Paragraph dialog 14-37

event procedure COPY file 3-8

Event tab 15-3
- screen form 14-6

events 15-3
- AfterDoPrint 17-19
- Before and After Print 17-18
- BeforeDoPrint 17-19
- descriptions of 14-29
- LoadGridInit 17-19
- LoadGridNext 17-19
- paged grid control 15-11
- populating reports with data 17-18

Execute command 22-4

Execute with Parameters command 22-4

executing a program 9-26

executing a remote program 9-26

executing a screen 14-41

Exit Debugger command 20-12, 22-6

external applications, linking to 4-33

external COPY files 4-25

extern-cpy tags 4-26

extern-def tags 4-24

F

file descriptor
- COPY files 4-24
- Data Designer options 4-29

File Designer 8-10
- changing a key 8-21
- data layout files from .fd and .sl files 8-4
- define a key 8-20
- Definition tab 8-13
- File Control tab 8-10
- import COPY file 8-17
- Key tab 8-20
- link COPY file 8-17
- removing a key 8-21
- XFD tab 8-24

file handling code 10-6
- in the Event Editor 8-30

File menu
- Close Workspace 6-6
- Open 12-2
- Open Project 6-10
- Page Setup 2-15, 11-10
- Print 2-14
- Print Preview 11-10
- Print Setup 2-14
- Save Workspace 6-4

file prefix 4-12

file record size, increasing Vision 19-16

file tags 4-24

file transfer utility 19-9

file type reference 3-10

File view 3-4

File/New dialog 12-3

file-control tags 4-24

files
- adding to a project 6-13
- closing 9-18
- converting indexed files 19-20
- creating 9-12
- creating with **vutil** 19-18
- deleting 9-19
- displaying b-tree 19-21
- extracting records from 19-19
- information extracted by **vutil** 19-17
- loading data 19-19
- managing 5-2
- moving 6-16
- opening 9-18
- placing in project directories 3-15
- properties 9-19
- rebuilding with **vutil** 19-17
- removing from a project 6-15
- reparsing 9-20
- resetting user count 19-17
- saving 9-18
- size information 19-19
- testing integrity of 19-17
- unloading data 19-19

filters, as print condition report property 17-12

Find
- regular expressions 21-5

Find command 21-2

Find in Files command 21-4

Find in Files dialog 21-4

Find in Objects command 21-8

Find Scope command 12-13

Font
- dialog 11-7, 24-19

footers 11-11

Format Code Editor options 4-15
Format menu
    ANSI to Terminal Format 12-4
    Capitalize 12-8
    Indent to Next 12-8
    Indent to Previous 12-8
    Lowercase 12-8
    Uppercase 12-8
frame control 15-8
    adding control first 15-8
    adding frame first 15-8
    copying and pasting controls 15-8
    grouping controls in the frame 15-8
    tab order 14-35, 15-9

G

General Code Editor options 4-15, 11-4
General Data Designer options 4-28
General Environment options 4-5
General Report Writer options 16-3
General Screen Designer options 13-3
Generate command 14-40, 22-4
Generate Document Code Generator options 4-20
generate files 14-40
generate screens 14-40
Generate STF Document command 14-40
generated file attributes 4-21
generating a program 9-21
generating native code 19-7
Go command 20-12, 22-6
Go To commands 12-6
Go to Definition command 12-16
Go to Next Error command 12-17

Go to Previous Error command 12-17
graphical controls
  resize layout manager 14-8, 14-28
graphical Screen Import Utility 24-30
grid control 15-9
  entry mode 15-10
  horizontal scrolling 15-9
  keys 15-10
  LoadGridInit 17-19
  LoadGridNext 17-19
  modes 15-10
  navigate mode 15-10
  paged 15-11
  rows, columns, and records 15-9
grid height option 13-3, 16-3
grid width option 13-3, 16-3
group header and footer 17-6

H

headers 11-11
help from Technical Support 1-6

I

I/O paragraph handling 4-22
icon file 9-8
Identification area 11-6
implementation tips, character screen import 24-28
-import runtime option 24-32
import COPY file 8-17
Import FD/SL Files dialog 8-7
Import Screen pop-up menu 24-33
import.out 24-33

importing a graphical screen 24-30, 24-33
importing character-based screens 24-11
    benefits and restrictions 24-12
Indent to Next command 12-8
Indent to Previous command 12-8
Indicator area 11-6
initial program property 9-8
insert data from file 12-9
Insert File command 12-9
Insert File dialog 12-9
Insert New Page command 15-16
Insert Separator command 15-25
inserting a code template 12-17
inserting bookmarks 12-7
installing AcuBench 2-6
integrated debugger 20-2
    tracing levels 20-11
Intel processor 19-8
Interrupt command 20-12, 22-6
IS-REMOTE flag 17-35

K

keyboard commands
    moving through text 12-6
    selecting text 12-6
Keyboard Environment options 4-11
keyboard shortcuts 23-2
    adding 4-11
    adding, Report Composer 16-8
    adding, Screen Designer 13-7
    Code Editor category 23-24
    Main category 23-3
    Screen Designer category 13-7, 23-30

keys 8-20
Keyword Code Editor options 4-15, 11-8
keyword set
  creating 11-9
  deleting 11-9
  modifying 11-9
  setting indent value 11-10
keywords
  adding and deleting 11-9

L

large applications
  and performance 3-20
  developing in AcuBench 3-20
  workspace organization 3-21
Launch toolbar 22-8
Layout Data property 14-9, 14-28
Layout Data Settings dialog 14-28
layout manager 14-28
  resize 14-8
Layout Manager property 14-9
license codes 2-6
Line Number pane, width of 11-4
link COPY file 8-17
Linkage Editor 10-12
  basic use of 10-12
Linkage options 4-29
Linkage section COPY file 3-9, 4-25
linkage tags 4-25
linkpara tags 4-25
list box control 15-12
List Constants command 12-14
List COPY Files command 12-16
list file directory 3-15

List Paragraphs command 12-14
List Variables command 12-14
List View command 14-10
LoadGridInit event 17-19
  code samples 17-22
  generated files 17-25
  modifying the source file 17-25
LoadGridNext event 17-19
  code samples 17-22
  described 17-22
  generated files 17-25
  modifying source file 17-25
loading a previous version's ".ini" file 2-7
locating variables in "Configuration file entries" categories 7-11
Lock Controls command 14-31
locking controls 14-31
logo screen 9-7
logutil 19-13
Lowercase command 12-8

M

macros 4-13
Main keyboard shortcuts 23-3
main screen 9-7
Make Same Size command (Width and/or Height) 14-31
Make XFD File command 8-31
mapping options
  Character-to-GUI Wizard 24-15
  Screen Import Utility 24-31
Memory command 20-7
Memory window 20-7, 22-6
menu bar 2-8
  changing 15-27
  creating 15-24

menu command list, version control 5-5
menu COPY file 3-8
Menu Designer 15-23
  creating a menu bar 15-24
  creating a pop-up menu 15-27
  dialog 15-24
  Insert Separator command 15-25
  Property window 15-24
menu items
  adding 15-24
  deleting 15-25
  moving 15-26
Microsoft ActiveX controls, licensing issues 15-22
minimum system requirements 2-6
Miscellaneous Environment options 4-13
mode 7-3
  Add/Remove Project Mode dialog 7-4
  selecting from project toolbar 22-4
modify code option 11-5
modifying breakpoints 20-17
modifying configuration variables 7-9
monitoring variables 20-18
moving a menu item 15-26
moving a tree view item 15-19
moving files 6-16
moving through text 12-6
multiple menu bars 15-27
multiple toolbars 15-30

N

Native Code command 19-8
native code, generating 19-7
navigate mode for a grid control 15-10

New command 22-2
New FD/SL dialog 8-5
New Folder dialog 6-12
New Program dialog 9-3
New Project command 6-7
New Screen dialog 14-3
Next Bookmark command 12-7
Next Page command 15-16
notation 2-5

O

object file directory 3-15
object library 7-16, 19-3
    creating 19-4
    creating with project management functions 7-16
    displaying information 19-6
    generating native code 19-7
    primary module 19-3
on-line documentation, accessing 2-3
Open command 12-2, 22-2
Open COPY File command 12-16
Open Project command 22-2
Open Workspace command 6-4
opening a new file 12-2
opening a screen 14-2
opening a workspace 6-4
opening an existing file 12-2
options
    loading 4-4
    mapping 24-16, 24-32
    saving 4-4
Options dialog box 4-2
Options/Code Editor dialog
    Keyword 4-15, 11-8

organizing the workspace 3-21
Output window 2-13, 20-2
Output Window command 22-2

P

page appearance 11-11
page header and footer properties 17-10
Page Setup command 2-15, 11-10
Page Setup dialog 2-15, 11-10
paged screen controls
  grid 15-11
  list box 15-13
Paragraph List box 12-15
  commands 12-15
Paragraph List Code Editor options 4-17
parameter completion 12-13
parsing 9-16
Paste command 12-5, 14-33, 22-2
pasting a control 14-33
pasting multiple controls
  changing variable values 14-33
phone numbers, for technical support 1-6
pop-up menus 15-27
Prefix Environment options 4-12
Previous Bookmark command 12-7
Previous Page command 15-16
prg-comment tags 4-23
prgid tags 4-23
primary module in object libraries 19-3
Print command 2-14, 22-2
print condition report property, as a filter 17-12
Print dialog 2-14
Print Preview command 11-10, 22-2

Print Preview screen 11-10
Print Setup command 2-14
Print Setup dialog 2-14
printing 2-14
  header and footer codes 2-15
  reports in thin client 17-32
Procedure COPY files 4-26
Procedure Division
  COPY file 3-9
  statements 14-30
product code and key 2-6
program
  compiling 9-21
  upgrading 24-4
program development approaches 3-12
program name
  changing 9-7
program properties
  description field 9-8
  generated files 9-9
  icon file 9-8
  initial program 9-8
  logo screen 9-7
  logo screen display time 9-7
  main screen 9-7
  prompt on exit 9-8
program structure file
  adding 9-4
  described 3-6
programs 3-6
  compiling to a remote server 9-22
  creating 9-2
  debugging 9-27
  executing 9-26
  generating 9-21

project 6-2
- development 3-2
- directories 3-14
- management 6-2
- prefix settings 3-14
- sample 2-5
- settings 7-2
- structure 6-2
- version control management 5-2

Project Settings
- Compiler tab 7-7
- dialog 7-2, 24-16, 24-31
- Environment tab 7-12
- Library tab 7-16
- object library 7-16
- Runtime tab 7-7

Project toolbar 22-4

projects
- adding COPY files to 9-16
- adding file to 6-13
- concurrent development 3-2, 6-7
- creating 6-7
- deleting 6-16
- described 3-5
- modes 7-3
- moving files between 6-16
- properties 6-10
- removing files from 6-15
- renaming 6-10

prompt on exit 9-8

properties 15-2
- common 15-2
- detail section 17-9
- group footer and header 17-9
- page footer and header 17-10

property values 15-3
report 17-9
report check box 18-7
report entry field 18-10
report footer 17-10
report header 17-10
report image 18-14
report label 18-16
report line 18-16
report occurs 18-17
report radio button 18-19
screen form 14-6
special properties 15-2
styles 15-3

Properties command 6-10
Properties dialog 24-17, 24-18
property items 15-3
property values 15-3
Property window 15-2
Property Window command 22-2
Property window Event tab 15-3
push button control 15-13

Q

Quick Watch command 22-7
Quick Watch window 20-6
Quit Debugging command 20-12, 22-6

R

radio button control 15-14
Rebuild command 22-4
Rebuild Workspace command 6-5, 9-24
Record Script command 20-14

record size, increasing for Vision files 19-16
Redo command 14-33, 22-2
Refresh command 12-16
refresh DLT 8-31
Regenerate Workspace command 6-6, 22-5
regular expressions 21-5
reinstalling AcuBench 2-7
remote compiler settings 9-22
removing a project mode 7-5
removing a value or comment from the configuration file 7-11
removing ActiveX controls 15-23
removing files 6-15
Reparse All command 9-20
Reparse command 9-20
Replace command 21-9
Replace dialog 21-9
Replace in Files dialog 21-11
Replace in Files function 21-10
Replace, regular expressions 21-5
report box 18-6
report check box 18-7
Report Component Toolbox
    report box 18-6
    report check box 18-7
    report entry field 18-10
    report grid 18-12
    report image 18-14
    report label 18-16
    report line 18-16
    report occurs 18-17
    report radio button 18-19
    report table 18-20
Report Component Toolbox command 22-3
Report Composer
    AfterDoPrint event 17-19

- Before and After print events 17-18
- BeforeDoPrint event 17-19
- creating a report 17-2
- detail section 17-6
- event paragraphs 17-18, 17-19
- generated files 17-23
- group footer and header 17-6
- LoadGridInit event 17-19
- LoadGridNext event 17-19
- page header and footer 17-6
- report box 18-6
- report check box 18-7
- report date time 18-8
- report entry field 18-10
- report grid 18-12
- report header and footer 17-6
- report image 18-14
- report label 18-16
- report line 18-16
- report occurs 18-17
- report Property window 17-9
- report radio button 18-19
- report table 18-20

report controls 17-13
- embedded spaces in 16-3

report date time 18-8

report entry field 18-10

report footer 17-6
- properties 17-11

report grid 18-12

report header 17-6
- properties 17-11

report image 18-14

report label 18-16

report line 18-16

report occurs control 18-17
report properties 17-9
report Property window 17-9
report radio button 18-19
report table 18-20
report templates 16-7
Report Writer options
    Auto alignment 16-3
    Collapse Spaces 16-3
    Default 16-5
    displaying property in Property window 16-6
    General 16-3
    grid properties 16-3
    setting default property values 16-4
    Unit of measure 16-3
    Visibility 16-6
reports
    creating 17-2
    graphical 17-31
    sample 2-5, 17-31
Reset Controls Tab Order dialog 14-34
resize layout manager 14-8, 14-28
Restart command 20-12, 22-6
restrictions
    Character-to-GUI Wizard 24-12
    Screen Import Utility 24-34
reusing a data layout file 8-29
Row Setting tab 18-23, 18-24
Run Script command 20-14
Run to Cursor command 20-14, 22-6
run-mainscr tags 4-26
RUNNAME environment variable 7-13
runtime configuration file 7-9
runtime debugger 20-2
runtime options, setting 7-7

S

sample project and programs 2-5
sample reports 2-5, 17-31
Save All command 22-2
Save command 22-2
Save Workspace command 6-4
saving a project 6-10
screen
  unique prefix 24-23
Screen Component Toolbox 2-12, 14-9
  ActiveX controls 15-20, 15-22
  List View command 14-10
screen controls 14-29, 15-2
  adding 14-9
  bar 15-5
  bitmap 15-5
  check box 15-6
  combo box 15-6
  date entry 15-7
  drawing 14-10
  entry field 15-8
  events 15-3
  frame 15-8
  grid 15-9
  list box 15-12
  paged grid 15-11
  paged list box 15-13
  properties 15-2
  push button 15-13
  radio button 15-14
  scroll bar 15-14
  selector 15-4
  standard 15-3
  status bar 15-15

tab 15-15
tab orientation 15-15
tree view 15-17
unique prefix 14-4
Web browser 15-20
screen conversion, character to graphical 24-11
screen COPY file 3-8, 4-25
screen description entry 14-29
Screen Designer
ActiveX controls 15-20
Character-to-GUI Wizard 24-21
drawing controls 14-10
Event Editor 14-35
keyboard shortcuts 13-7, 23-30
Menu Designer 15-23
Screen Import Utility 24-30
standard controls 15-3
Toolbar Designer 15-29
Screen Designer options
Auto alignment 13-3
Default 13-4
displaying property in Property window 13-5
General 13-3
grid height and width 13-3
setting default property values 13-4
snap to grid 13-3
Visibility 13-5
screen form
adding controls to 14-9
Event Tab 14-6
properties 14-6
screen import
code integration 24-23
implementation tips 24-28
Screen Import command 24-32

Screen Import Utility 24-30
- -import runtime option 24-32
- mapping options 24-31
- restrictions 24-34
- -Zw compiler option 24-31

screen template 13-6
- adding to New Screen dialog 13-6
- changing prefix 14-40
- sharing 13-7, 16-8

screen templates
- creating 14-40

screens
- creating 14-2
- unique prefix 14-4

scroll bar control 15-14
search string operators 21-5
searching a configuration file 7-11
SELECT statement COPY files 4-24
SELECT statement, defining 8-10
selecting text 12-6
selector 15-4
Send to Back command 14-35
Send to Front command 14-35
Sequence Number Area 11-6
server platforms supported 3-25
Set Active Mode command 22-4
setting a monitor 20-5
setting ACCEPT-CONTROL 4-22
setting ActiveX properties 15-22
setting and removing breakpoints 20-15
setting custom tabs 12-10
setting default property values 13-4, 16-4
setting headers and footers 11-11
setting or changing a comment on a variable 7-11
setting or changing the value of a variable 7-10

setting page appearance 11-11
setting program tags 4-27
Settings command 22-4
shortcut keys
    assigning 4-11
    removing 4-12
    Report Composer 16-8
    restoring defaults 4-12
    Screen Designer 13-7
    viewing definitions of 4-11
Skip to Line command 20-13, 22-6
snap to grid option 13-3, 16-3
sorting lists (Constant, COPY File, Paragraph, and Variable) 4-17
source file
    creating 9-12
    reparsing 9-20
source format options 11-5, 11-8
Space Across command 14-31
Space Down command 14-31
Sparc processor 19-8
Special Names COPY file 4-23
special properties 15-2
Split command 12-8
splitter 12-8
Stack Info window 20-6
standard controls 14-10, 15-3
Standard toolbar 22-2
status bar 2-13
status bar control 15-4, 15-15
Step Into command 20-13, 22-6
Step Out Paragraph command 20-13, 22-6
Step Out Program command 20-13, 22-6
Step Over command 20-13, 22-6
Stop Build command 6-6, 22-4
stopping a build 6-6

Structure view 3-4
styles 15-3
submenu items, adding, deleting, or moving 15-25
support, technical 1-6
switching threads in the debugger 20-19
system requirements 2-6

T

tab control 15-15
  Insert New Page command 15-16
  Next Page command 15-16
  orientation 15-15
  Previous Page command 15-16
  Send to Back command 14-35
  Send to Front command 14-35
Tab Order command 14-34
tab order, with frames 14-35
Tab Orientation property 15-15
tab size function 11-8
Table Setting dialog 18-23
  Cell Setting tab 18-21, 18-25
  Column Setting tab 18-23
  Row Setting tab 18-23, 18-24
Tabs Code Editor options 4-15
tabs function 12-10
tags
  adding 4-26
  described 4-22
technical support 1-6
Template Environment options 4-7
templates, source code 12-17
text block handling 12-10, 12-11
text navigation 12-6

thin client
    platform support 3-25
    printing graphical reports 17-32
THINNAME environment variable 7-13
Threads command 20-19
Threads dialog 20-19
Threads window 22-6
title bar 2-8
Toggle Breakpoint command 22-6
Toggle Guide command 14-32
Toolbar Designer 15-29
toolbars 2-8
    Align 22-8
    changing 15-30
    controlling 2-9
    creating 15-29
    Debug 22-6
    Editor 22-6
    hiding and displaying 2-10
    Launch 22-8
    moving 2-10
    multiple 15-30
    Project 22-4
    resizing 2-10
    Standard 22-2
toolbars, AcuBench
    adding buttons to 4-32
    creating 4-31
    customizing 4-31
    deleting 4-31
    removing buttons from 4-32
    spacers 4-32
Tools menu
    Acu4glFD 19-21
    Axdefgen 15-21

  Options 4-2
  Xml2FD 19-22
Tools/Options Load command 2-7
Trace File command 20-10
Trace Flush command 20-10
Trace General command 20-10
Trace Level dialog 20-11
Trace Levels command 20-11
Trace Option command 20-10
Trace Paragraphs command 20-10
Trace Screens command 20-10
Trace to Debug Window command 20-11
tracing levels, integrated debugger 20-11
transaction logs 19-13
tree view control 15-17
Tree View Designer 15-17
tree view item
  adding 15-18
  deleting 15-19
  moving 15-19

U

Undo command 14-33, 22-2
uninstalling AcuBench 2-7
unlocking controls 14-31
upgrading a program file 24-4
Uppercase command 12-8
Use AcuXUI 14-42
Use Remote Compile 9-22
Use Thin Client command 22-5
user interface
  customizing 4-2
  development window 2-12

- menu bar 2-8
- Output window 2-13
- overview 2-7
- Screen Component Toolbox 2-12
- status bar 2-13
- title bar 2-8
- toolbars 2-8
- Workspace window 2-11

User's Guide, accessing on-line 2-3
using bookmarks 12-7
using tabs 12-10
utilities 19-2
- Boomerang 9-24

V

Variable List box 12-15
- commands 12-15

Variable List Code Editor options 4-17
Verb Block Match command 12-13
version control
- adding commands 5-8
- changing the order of commands 5-9
- commands 5-5
- deleting commands 5-9
- fundamental source control commands 5-6
- managing files and projects 5-2
- menu command list 5-5
- modifying commands 5-9
- options window fields 5-6
- overview 5-2
- saving commands 5-9
- third party software 5-2

version control arguments
- Comment argument 5-10
- Current Directory argument 5-10
- File Directory argument 5-10
- File Name argument 5-10
- File Path argument 5-10
- Folder Name argument 5-10
- Project Directory argument 5-10
- Project Name argument 5-10
- Target Directory argument 5-10
- Workspace Name argument 5-10
- Workspace Path argument 5-10

Version Control Environment options 4-8

version control fields
- Arguments field 5-7
- Ask Argument option 5-7
- Command field 5-7
- Use Output Window 5-7
- Working Directory argument 5-7

version control interface 5-5

vertical block select function 11-4, 12-5

View Grid command 14-32

viewing breakpoints 20-17

vio utility 19-9
- input mode 19-12
- output mode 19-10

virtual space 11-4

Visibility Report Writer options 16-6

Visibility Screen Designer options 13-5

Vision files, increasing record size of 19-16

vutil (Vision File Utility) 19-15
- converting indexed file 19-20
- creating a file 19-18
- extracting a record 19-19
- file information 19-17

- file integrity 19-17
- file size 19-19
- listing b-tree 19-21
- loading data 19-19
- rebuilding 19-17
- unloading data 19-19
- user count 19-17

vutil augment function 19-16

W

Watch command 20-5
Watch window 20-4, 22-6
Web browser control 15-20
Window menu, Split 12-8
working directory macros 4-13
Working Storage Editor 10-11
- basic use 10-12

working with reports 17-2
Working-Storage COPY file 3-9, 4-25
Working-Storage options 4-29
workspace 3-3, 6-2
- building 6-4
- closing 6-6
- creating 6-3
- Data View 3-5
- file 3-3, 3-5
- File view 3-4
- opening 6-4
- organization 3-21
- rebuilding 6-5
- regenerating 6-6, 22-5
- saving 6-4
- Structure view 3-4

Workspace command 22-2
Workspace window 2-11, 3-5, 20-2
    Structure view 3-4

X

XFD files 8-24
    creating in AcuBench 8-25, 8-31
    NAME directive 8-26
    use in AcuBench 8-26
xml2fd utility 19-22
XUIJAR environment variable 14-42

Z

-Zw compiler option
    Character-to-GUI Wizard 24-15
    Screen Import Utility 24-31