

# Progress Basic Development Tools

© 2001 Progress Software Corporation. All rights reserved.

Progress® software products are copyrighted and all rights are reserved by Progress Software Corporation. This manual is also copyrighted and all rights are reserved. This manual may not, in whole or in part, be copied, photocopied, translated, or reduced to any electronic medium or machine-readable form without prior consent, in writing, from Progress Software Corporation.

The information in this manual is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear in this document.

The references in this manual to specific platforms supported are subject to change.

Progress, Progress Results, Provision and WebSpeed are registered trademarks of Progress Software Corporation in the United States and other countries. Apptivity, AppServer, ProVision Plus, SmartObjects, IntelliStream, and other Progress product names are trademarks of Progress Software Corporation.

SonicMQ is a trademark of Sonic Software Corporation in the United States and other countries.

Progress Software Corporation acknowledges the use of Raster Imaging Technology copyrighted by Snowbound Software 1993-1997 and the IBM XML Parser for Java Edition.

© IBM Corporation 1998-1999. All rights reserved. U.S. Government Users Restricted Rights — Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Progress is a registered trademark of Progress Software Corporation and is used by IBM Corporation in the mark Progress/400 under license. Progress/400 AND 400® are trademarks of IBM Corporation and are used by Progress Software Corporation under license.

Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Any other trademarks and/or service marks contained herein are the property of their respective owners.

.

May 2001




Product Code: 4506
Item Number: 81073;9.1C

# Contents


**Preface** . . . . . **ix**
Purpose . . . . . ix
Audience . . . . . ix
Organization of This Manual . . . . . ix
Typographical Conventions . . . . . x
Syntax Notation . . . . . xi
Example Procedures . . . . . xiv
Progress Messages . . . . . xvi
Other Useful Documentation . . . . . xvii
Getting Started . . . . . xvii
Development Tools . . . . . xix
Reporting Tools . . . . . xx
4GL . . . . . xx
Database . . . . . xxi
DataServers . . . . . xxi
SQL-89/Open Access . . . . . xxii
SQL-92 . . . . . xxii
Deployment . . . . . xxiii
WebSpeed . . . . . xxiii
Reference . . . . . xxiv

**1. Application Development Environment** . . . . . **1–1**
1.1 Starting the ADE Tools . . . . . 1–2
1.1.1 The Procedure Editor . . . . . 1–2
1.1.2 The Application Compiler . . . . . 1–3
1.2 Accessing Menus and Menu Options . . . . . 1–3
1.3 Navigating in Dialog Boxes and Windows . . . . . 1–4

1.4 Using the Tools Menu . . . . . 1–4
1.4.1 Tools→ Procedure Editor . . . . . 1–4
1.4.2 Tools→ Data Dictionary . . . . . 1–5
1.4.3 Tools→ OS Shell . . . . . 1–5
1.4.4 Tools→ Application Compiler . . . . . 1–5
1.5 Using the Help Menu . . . . . 1–5
1.5.1 Help→ Messages . . . . . 1–5
1.5.2 Help→ Recent Messages . . . . . 1–6
1.5.3 Help→ Keyboard . . . . . 1–6
1.5.4 Help→ About *Tool* . . . . . 1–6

**2. Procedure Editor Tasks . . . . . 2–1**
2.1 Starting the Procedure Editor . . . . . 2–2
2.2 Using Edit Buffers . . . . . 2–4
2.2.1 Listing Open Buffers . . . . . 2–5
2.2.2 Switching Buffers . . . . . 2–5
2.2.3 Displaying Buffer Information . . . . . 2–5
2.3 Working with Procedures . . . . . 2–6
2.3.1 Creating a New Procedure . . . . . 2–6
2.3.2 Opening a Procedure . . . . . 2–6
2.3.3 Saving a Procedure . . . . . 2–7
2.3.4 Printing a Procedure . . . . . 2–8
2.3.5 Closing a Procedure . . . . . 2–8
2.4 Editing Text . . . . . 2–8
2.4.1 Entering Text . . . . . 2–9
2.4.2 Selecting Text . . . . . 2–10
2.4.3 Cutting, Copying, and Pasting Text . . . . . 2–10
2.4.4 Searching for Text . . . . . 2–10
2.4.5 Inserting a File into a Procedure . . . . . 2–13
2.4.6 Inserting a Field Name into a Procedure . . . . . 2–13
2.5 Compiling, Running, and Checking Procedures . . . . . 2–14
2.5.1 Checking a Procedure's Syntax . . . . . 2–14
2.5.2 Compiling and Running a Procedure . . . . . 2–14
2.5.3 Debugging a Procedure . . . . . 2–15
2.6 Exiting the Procedure Editor . . . . . 2–15

**3. Procedure Editor Integration Hooks . . . . . 3–1**
3.1 ADE Event (adecomm/_adeevnt.p) . . . . . 3–3
3.2 Parameters . . . . . 3–3
3.2.1 Input Parameters . . . . . 3–3
3.2.2 Output Parameters . . . . . 3–4
3.3 Events . . . . . 3–4
3.3.1 File-specific Events . . . . . 3–4
3.3.2 STARTUP and SHUTDOWN Events . . . . . 3–6
3.4 Usage . . . . . 3–6

**4. Procedure Editor Reference . . . . . 4–1**
4.1 Procedure Editor Menu Bar . . . . . 4–2
4.2 File Menu . . . . . 4–2
4.2.1 File→ New . . . . . 4–3
4.2.2 File→ Open . . . . . 4–3
4.2.3 File→ Close . . . . . 4–5
4.2.4 File→ Save . . . . . 4–5
4.2.5 File→ Save As . . . . . 4–5
4.2.6 File→ Print . . . . . 4–6
4.2.7 File→ Exit . . . . . 4–7
4.3 Edit Menu . . . . . 4–7
4.3.1 Edit→ Cut . . . . . 4–8
4.3.2 Edit→ Copy . . . . . 4–8
4.3.3 Edit→ Paste . . . . . 4–8
4.3.4 Edit→ Insert File . . . . . 4–8
4.3.5 Edit→ Insert Fields . . . . . 4–9
4.4 Search Menu . . . . . 4–10
4.4.1 Search→ Find . . . . . 4–11
4.4.2 Search→ Find Next . . . . . 4–11
4.4.3 Search→ Find Previous . . . . . 4–12
4.4.4 Search→ Replace . . . . . 4–12
4.4.5 Search→ Goto Line . . . . . 4–13
4.5 Buffer Menu . . . . . 4–13
4.5.1 Buffer→ List . . . . . 4–14
4.5.2 Buffer→ Next Buffer . . . . . 4–14
4.5.3 Buffer→ Previous Buffer . . . . . 4–14
4.5.4 Buffer→ Information . . . . . 4–15
4.6 Compile Menu . . . . . 4–16
4.6.1 Compile→ Run . . . . . 4–16
4.6.2 Compile→ Check Syntax . . . . . 4–18
4.6.3 Compile→ Debug . . . . . 4–18
4.6.4 Compile→ Compiler Messages . . . . . 4–18
4.7 Tools Menu . . . . . 4–18
4.8 Help Menu . . . . . 4–18

**5. Application Compiler . . . . . 5–1**
5.1 Starting the Application Compiler . . . . . 5–2
5.2 Using the Files/Directories Selection List . . . . . 5–2
5.3 Using the Toggle Boxes . . . . . 5–2
5.4 Using the Action Buttons . . . . . 5–4
5.5 Using the Menu Bar . . . . . 5–6
5.5.1 File Menu . . . . . 5–6
5.5.2 Compile Menu . . . . . 5–6
5.5.3 Tools Menu . . . . . 5–7
5.5.4 Options Menu . . . . . 5–7
5.5.5 Help Menu . . . . . 5–11
5.6 Compiling Source Files . . . . . 5–12

**Index . . . . . Index–1**

**Figures**


Figure 2–1: Procedure Editor Window . . . . . 2–3
Figure 2–2: Buffer List Dialog Box . . . . . 2–5
Figure 4–1: Procedure Editor Menu Bar . . . . . 4–2
Figure 4–2: Open Dialog Box . . . . . 4–3
Figure 4–3: Files Dialog Box . . . . . 4–4
Figure 4–4: Close Alert Box . . . . . 4–5
Figure 4–5: Save As Dialog Box . . . . . 4–6
Figure 4–6: Save Buffers with Changes Dialog Box . . . . . 4–7
Figure 4–7: Insert File Dialog Box . . . . . 4–9
Figure 4–8: Field Selector Dialog Box . . . . . 4–9
Figure 4–9: Find Dialog Box . . . . . 4–11
Figure 4–10: Replace Dialog Box . . . . . 4–12
Figure 4–11: Goto Line Dialog Box . . . . . 4–13
Figure 4–12: Buffer List Dialog Box . . . . . 4–14
Figure 4–13: Buffer Information Dialog Box . . . . . 4–15
Figure 4–14: Example Procedure Output to Screen . . . . . 4–17
Figure 4–15: Compiler Messages Dialog Box . . . . . 4–18
Figure 5–1: Progress Application Compiler Window . . . . . 5–2
Figure 5–2: File Specification Dialog Box . . . . . 5–5
Figure 5–3: Compiler Results Dialog Box . . . . . 5–7
Figure 5–4: Compiler Options Dialog Box . . . . . 5–8

**Tables**


Table 1–1: Tools Menu . . . . . 1–4
Table 1–2: Help Menu . . . . . 1–5
Table 2–1: Procedure Editor Features . . . . . 2–3
Table 3–1: File-specific Events . . . . . 3–4
Table 3–2: Usage Scenario for the Procedure Editor . . . . . 3–7
Table 4–1: Procedure Editor Menus . . . . . 4–2
Table 4–2: File Menu . . . . . 4–2
Table 4–3: Files Dialog Box . . . . . 4–4
Table 4–4: Save As Dialog Box Fields . . . . . 4–6
Table 4–5: Edit Menu . . . . . 4–7
Table 4–6: Field Selector Dialog Box . . . . . 4–10
Table 4–7: Search Menu . . . . . 4–10
Table 4–8: Find Dialog Box . . . . . 4–11
Table 4–9: Replace Dialog Box . . . . . 4–12
Table 4–10: Buffer Menu . . . . . 4–13
Table 4–11: Buffer Information Dialog Box . . . . . 4–15
Table 4–12: Compile Menu . . . . . 4–16
Table 5–1: The Save New .r Toggle Box and the Compilation Process . . . . . 5–3
Table 5–2: Compiler Action Buttons . . . . . 5–4
Table 5–3: File Specification Dialog Box . . . . . 5–5
Table 5–4: Application Compiler Menu . . . . . 5–6
Table 5–5: File Menu . . . . . 5–6
Table 5–6: Compile Menu . . . . . 5–6
Table 5–7: Options Menu . . . . . 5–7
Table 5–8: Compiler Options Dialog Box . . . . . 5–8

# Preface

## Purpose

This book is a user guide for the Progress Version 9 basic development toolset for character mode. These tools include the Progress Procedure Editor, the Data Dictionary, and the Application Compiler.

## Audience

This book is intended for developers who want to use the Progress basic development tools to develop their applications.

## Organization of This Manual

Chapter 1, "Application Development Environment"

Describes how to access the Application Development Environment (ADE), how to access each of the Progress tools, and how to use on-line help for error messages.

Chapter 2, "Procedure Editor Tasks"

Describes how to access the Procedure Editor and how to use it to perform tasks. The Procedure Editor allows you to create, write, compile, and run Progress procedures.

Chapter 3, "Procedure Editor Integration Hooks"

Describes the integration hooks you can add to the Procedure Editor.

Chapter 4, "Procedure Editor Reference"

Describes the Procedure Editor menu options and dialog boxes.

Chapter 5, “Application Compiler”

Describes how to access the Application Compiler and how to use each of the Application Compiler menu options. The Application Compiler allows you to compile a set of source procedures and create Progress r-code.

## Typographical Conventions

This manual uses the following typographical conventions:

- **Bold typeface** indicates:
  - Commands or characters that the user types
  - That a word carries particular weight or emphasis
- *Italic typeface* indicates:
  - Progress variable information that the user supplies
  - New terms
  - Titles of complete publications
- `Monospaced typeface` indicates:
  - Code examples
  - System output
  - Operating system filenames and pathnames

The following typographical conventions are used to represent keystrokes:

- Small capitals are used for Progress key functions and generic keyboard keys.

  **END-ERROR, GET, GO**
  **ALT, CTRL, SPACEBAR, TAB**

- When you have to press a combination of keys, they are joined by a dash. You press and hold down the first key, then press the second key.

  **CTRL-X**

- When you have to press and release one key, then press another key, the key names are separated with a space.

  **ESCAPE H**
  **ESCAPE CURSOR-LEFT**

## Syntax Notation

The syntax for each component follows a set of conventions:

- Uppercase words are keywords. Although they are always shown in uppercase, you can use either uppercase or lowercase when using them in a procedure.

  In this example, ACCUM is a keyword:

  **SYNTAX**

  ```
  ACCUM aggregate expression
  ```

- Italics identify options or arguments that you must supply. These options can be defined as part of the syntax or in a separate syntax identified by the name in italics. In the ACCUM function above, the *aggregate* and *expression* options are defined with the syntax for the ACCUM function in the *Progress Language Reference*.

- You must end all statements (except for DO, FOR, FUNCTION, PROCEDURE, and REPEAT) with a period. DO, FOR, FUNCTION, PROCEDURE, and REPEAT statements can end with either a period or a colon, as in this example:

  ```
  FOR EACH Customer:
     DISPLAY Name.
  END.
  ```

- Square brackets ( **[ ]** ) around an item indicate that the item, or a choice of one of the enclosed items, is optional.

  In this example, STREAM *stream*, UNLESS-HIDDEN, and NO-ERROR are optional:

  **SYNTAX**

  ```
  DISPLAY [ STREAM stream ] [ UNLESS-HIDDEN ] [ NO-ERROR ]
  ```

In some instances, square brackets are not a syntax notation, but part of the language.

For example, this syntax for the INITIAL option uses brackets to bound an initial value list for an array variable definition. In these cases, normal text brackets ( [ ] ) are used:

**SYNTAX**

```
INITIAL [ constant [ , constant ] ... ]
```

**NOTE:** The ellipsis ( **. . .** ) indicates repetition, as shown in a following description.

- Braces ( **{ }** ) around an item indicate that the item, or a choice of one of the enclosed items, is required.

  In this example, you must specify the items BY and *expression* and can optionally specify the item DESCENDING, in that order:

  **SYNTAX**

  ```
  { BY expression [ DESCENDING ] }
  ```

  In some cases, braces are not a syntax notation, but part of the language.

  For example, a called external procedure must use braces when referencing arguments passed by a calling procedure. In these cases, normal text braces ( { } ) are used:

  **SYNTAX**

  ```
  { &argument-name }
  ```

- A vertical bar ( **|** ) indicates a choice.

  In this example, EACH, FIRST, and LAST are optional, but you can only choose one:

  **SYNTAX**

  ```
  PRESELECT [ EACH | FIRST | LAST ] record-phrase
  ```

In this example, you must select one of *logical-name* or *alias*:

**SYNTAX**

```
CONNECTED ( { logical-name | alias } )
```

- Ellipses ( **. . .** ) indicate that you can choose one or more of the preceding items. If a group of items is enclosed in braces and followed by ellipses, you must choose one or more of those items. If a group of items is enclosed in brackets and followed by ellipses, you can optionally choose one or more of those items.

  In this example, you must include two expressions, but you can optionally include more. Note that each subsequent expression must be preceded by a comma:

  **SYNTAX**

  ```
  MAXIMUM ( expression , expression [ , expression ] ... )
  ```

  In this example, you must specify MESSAGE, then at least one of *expression* or SKIP, but any additional number of *expression* or SKIP is allowed:

  **SYNTAX**

  ```
  MESSAGE { expression | SKIP [ (n) ] } ...
  ```

  In this example, you must specify {*include-file*, then optionally any number of *argument* or &*argument-name* = "*argument-value*", and then terminate with }:

  **SYNTAX**

  ```
  { include-file
      [ argument | &argument-name = "argument-value" ] ... }
  ```

- In some examples, the syntax is too long to place in one horizontal row. In such cases, **optional** items appear individually bracketed in multiple rows in order, left-to-right and top-to-bottom. This order generally applies, unless otherwise specified. **Required** items also appear on multiple rows in the required order, left-to-right and top-to-bottom. In cases where grouping and order might otherwise be ambiguous, braced (required) or bracketed (optional) groups clarify the groupings.

In this example, WITH is followed by several optional items:

**SYNTAX**

```
WITH [ ACCUM max-length ] [ expression DOWN ]
[ CENTERED ] [ n COLUMNS ] [ SIDE-LABELS ]
[ STREAM-IO ]
```

In this example, ASSIGN requires one of two choices: either one or more of *field*, or one of *record*. Other options available with either *field* or *record* are grouped with braces and brackets. The open and close braces indicate the required order of options:

**SYNTAX**

```
ASSIGN {   { [ FRAME frame ]
               { field [ = expression ] }
               [ WHEN expression ]
           } ...
         | { record [ EXCEPT field ... ] }
       }
```

## Example Procedures

This manual provides numerous example procedures that illustrate syntax and concepts. Examples use the following conventions:

- They appear in boxes with borders.
- If they are available online, the name of the procedure appears above the left corner of the box and starts with a prefix associated with the manual that references it, as follows:
  - `e-` — *Progress External Program Interfaces*, for example, `e-ddeex1.p`
  - `lt-` — *Progress Language Tutorial*, for example, `lt-05-s3.p`
  - `p-` — *Progress Programming Handbook*, for example, `p-br01.p`
  - `r-` — *Progress Language Reference*, for example, `r-dynbut.p`

If the name does not start with a listed prefix, the procedure is not available online.

- If they are not available online, they compile as shown, but might not execute for lack of completeness.

### Accessing Files in Procedure Libraries

Documentation examples are stored in procedure libraries, `prodoc.pl` and `prohelp.pl`, in the `src` directory where Progress is installed.

You must first create all subdirectories required by a library before attempting to extract files from the library. You can see what directories and subdirectories a library needs by using the PROLIB -list command to view the contents of the library. See the *Progress Client Deployment Guide* for more details on the PROLIB utility.

Extracting source files from a procedure library involves running PROENV to set up your Progress environment, creating the directory structure for the files you want to extract, and running PROLIB.

### Extracting Source Files from Procedure Libraries on UNIX Platforms

To extract `p-wrk1.p` from `prodoc.pl`, a procedure library, follow these steps at the UNIX system prompt:

1 ♦ Run the PROENV utility:

```
install-dir/dlc/bin/proenv
```

Running proenv sets the DLC environment variable to the directory where you installed Progress (by default, `/usr/dlc`). The proenv utility also adds the DLC environment variable to your PATH environment variable and adds the bin directory (`PATH=%DLC%;%DLC%\bin;%PATH%`).

2 ♦ At the proenv prompt, create the `prodoc` directory in your Progress working directory:

```
mkdir prodoc
```

3 ♦ Create the `proghand` directory under `prodoc`:

```
mkdir prodoc/proghand
```

4 ♦ To extract all examples in a procedure library directory, run the PROLIB utility:

```
prolib $DLC/src/prodoc.pl -extract prodoc/proghand/*.*
```

PROLIB extracts all examples into `prodoc\langref`.

To extract one example, run PROLIB and specify the file that you want to extract as it is stored in the procedure library:

```
prolib $DLC/src/prodoc.pl -extract prodoc/proghand/p-wrk-1.p
```

PROLIB extracts `p-wrk-1.p` into `prodoc/proghand`.

## Progress Messages

Progress displays several types of messages to inform you of routine and unusual occurrences:

- Execution messages inform you of errors encountered while Progress is running a procedure (for example, if Progress cannot find a record with a specified index field value).
- Compile messages inform you of errors found while Progress is reading and analyzing a procedure prior to running it (for example, if a procedure references a table name that is not defined in the database).
- Startup messages inform you of unusual conditions detected while Progress is getting ready to execute (for example, if you entered an invalid startup parameter).

After displaying a message, Progress proceeds in one of several ways:

- Continues execution, subject to the error-processing actions that you specify, or that are assumed, as part of the procedure. This is the most common action taken following execution messages.
- Returns to the Progress Procedure Editor so that you can correct an error in a procedure. This is the usual action taken following compiler messages.
- Halts processing of a procedure and returns immediately to the Procedure Editor. This does not happen often.
- Terminates the current session.

Progress messages end with a message number in parentheses. In this example, the message number is 200:

```
** Unknown table name table. (200)
```

You can use the Progress PRO command to start a single-user mode character Progress client session and view a brief description of a message by providing its number. Follow these steps:

1 ♦ Start the Progress Procedure Editor:

```
install-dir/dlc/bin/pro
```

2 ♦ Press **F3** to access the menu bar, then choose Help→ Messages.

3 ♦ Type the message number, and press **ENTER**. Details about that message number appear.

4 ♦ Press **F4** to close the message, press **F3** to access the Procedure Editor menu, and choose File→ Exit.

## Other Useful Documentation

This section lists Progress Software Corporation documentation that you might find useful. Unless otherwise specified, these manuals support both Windows and Character platforms and are provided in electronic documentation format on CD-ROM.

### Getting Started

*Progress Electronic Documentation Installation and Configuration Guide* (Hard copy only)

A booklet that describes how to install the Progress EDOC viewer and collection on UNIX and Windows.

*Progress Installation and Configuration Guide Version 9 for UNIX*

A manual that describes how to install and set up Progress Version 9.1 for the UNIX operating system.

*Progress Installation and Configuration Guide Version 9 for Windows*

A manual that describes how to install and set up Progress Version 9.1 for all supported Windows and Citrix MetaFrame operating systems.

*Progress Version 9 Product Update Bulletin*

A guide that provides a brief description of each new feature of the release. The booklet also explains where to find more detailed information in the documentation set about each new feature.

*Progress Application Development Environment — Getting Started* (Windows only)

A practical guide to graphical application development within the Progress Application Development Environment (ADE). This guide includes an overview of the ADE and its tools, an overview of Progress SmartObject technology, and tutorials and exercises that help you better understand SmartObject technology and how to use the ADE to develop applications.

*Progress Language Tutorial for Windows* and *Progress Language Tutorial for Character*

Platform-specific tutorials designed for new Progress users. The tutorials use a step-by-step approach to explore the Progress application development environment using the 4GL.

*Progress Master Glossary for Windows* and *Progress Master Glossary for Character* (EDOC only)

Platform-specific master glossaries for the Progress documentation set. These books are in electronic format only.

*Progress Master Index and Glossary for Windows* and *Progress Master Index and Glossary for Character* (Hard copy only)

Platform-specific master indexes and glossaries for the Progress hard-copy documentation set.

*Progress Startup Command and Parameter Reference*

A reference manual that describes the Progress startup commands and parameters in alphabetical order.

*Welcome to Progress* (Hard copy only)

A booklet that explains how Progress software and media are packaged. An icon-based map groups the documentation by functionality, providing an overall view of the documentation set. *Welcome to Progress* also provides descriptions of the various services Progress Software Corporation offers.

### Development Tools

*Progress ADM 2 Guide*

A guide to using the Application Development Model, Version 2 (ADM 2) application architecture to develop Progress applications. It includes instructions for building and using Progress SmartObjects.

*Progress ADM 2 Reference*

A reference for the Application Development Model, Version 2 (ADM 2) application. It includes descriptions of ADM 2 functions and procedures.

*Progress AppBuilder Developer's Guide* (Windows only)

A programmer's guide to using the Progress AppBuilder visual layout editor. AppBuilder is a Rapid Application Development (RAD) tool that can significantly reduce the time and effort required to create Progress applications.

*Progress Basic Database Tools* (Character only; information for Windows is in online help)

A guide for the Progress Database Administration tools, such as the Data Dictionary.

*Progress Debugger Guide*

A guide for the Progress Application Debugger. The Debugger helps you trace and correct programming errors by allowing you to monitor and modify procedure execution as it happens.

*Progress Help Development Guide* (Windows only)

A guide that describes how to develop and integrate an online help system for a Progress application.

*Progress Translation Manager Guide* (Windows only)

A guide that describes how to use the Progress Translation Manager tool to manage the entire process of translating the text phrases in Progress applications.

*Progress Visual Translator Guide* (Windows only)

A guide that describes how to use the Progress Visual Translator tool to translate text phrases from procedures into one or more spoken languages.

### Reporting Tools

*Progress Report Builder Deployment Guide* (Windows only)

An administration and development guide for generating Report Builder reports using the Progress Report Engine.

*Progress Report Builder Tutorial* (Windows only)

A tutorial that provides step-by-step instructions for creating eight sample Report Builder reports.

*Progress Report Builder User's Guide* (Windows only)

A guide for generating reports with the Progress Report Builder.

*Progress Results Administration and Development Guide* (Windows only)

A guide for system administrators that describes how to set up and maintain the Results product in a graphical environment. This guide also describes how to program, customize, and package Results with your own products. In addition, it describes how to convert character-based Results applications to graphical Results applications.

*Progress Results User's Guide for Windows* and *Progress Results User's Guide for UNIX*

Platform-specific guides for users with little or no programming experience that explain how to query, report, and update information with Results. Each guide also helps advanced users and application developers customize and integrate Results into their own applications.

### 4GL

*Building Distributed Applications Using the Progress AppServer*

A guide that provides comprehensive information about building and implementing distributed applications using the Progress AppServer. Topics include basic product information and terminology, design options and issues, setup and maintenance considerations, 4GL programming details, and remote debugging.

*Progress External Program Interfaces*

A guide to accessing non-Progress applications from Progress. This guide describes how to use system clipboards, UNIX named pipes, Windows dynamic link libraries, Windows dynamic data exchange, Windows ActiveX controls, and the Progress Host Language Call Interface to communicate with non-Progress applications and extend Progress functionality.

*Progress Internationalization Guide*

A guide to developing Progress applications for markets worldwide. The guide covers both internationalization—writing an application so that it adapts readily to different locales (languages, cultures, or regions)—and localization—adapting an application to different locales.

*Progress Language Reference*

A three-volume reference set that contains extensive descriptions and examples for each statement, phrase, function, operator, widget, attribute, method, and event in the Progress language.

*Progress Programming Handbook*

A two-volume handbook that details advanced Progress programming techniques.

### Database

*Progress Database Design Guide*

A guide that uses a sample database and the Progress Data Dictionary to illustrate the fundamental principles of relational database design. Topics include relationships, normalization, indexing, and database triggers.

*Progress Database Administration Guide and Reference*

This guide describes Progress database administration concepts and procedures. The procedures allow you to create and maintain your Progress databases and manage their performance.

### DataServers

Progress DataServer Guides

These guides describe how to use the DataServers to access non-Progress databases. They provide instructions for building the DataServer modules, a discussion of programming considerations, and a tutorial. Each DataServer has its own guide, for example, the *Progress DataServer for ODBC Guide*, the *Progress DataServer for ORACLE Guide*, or the *Progress/400 Product Guide*.

MERANT ODBC Branded Driver Reference

The Enterprise DataServer for ODBC includes MERANT ODBC drivers for all the supported data sources. For configuration information, see the MERANT documentation, which is available as a PDF file in *installation-path*\odbc. To read this file you must have the Adobe Acrobat Reader Version 3.1 or higher installed on your system. If you do not have the Adobe Acrobat Reader, you can download it from the Adobe Web site at: http://www.adobe.com/prodindex/acrobat/readstep.html.

### SQL-89/Open Access

*Progress Embedded SQL-89 Guide and Reference*

A guide to Progress Embedded SQL-89 for C, including step-by-step instructions on building ESQL-89 applications and reference information on all Embedded SQL-89 Preprocessor statements and supporting function calls. This guide also describes the relationship between ESQL-89 and the ANSI standards upon which it is based.

*Progress Open Client Developer's Guide*

A guide that describes how to write and deploy Java and ActiveX applications that run as clients of the Progress AppServer. The guide includes information about how to expose the AppServer as a set of Java classes or as an ActiveX server.

*Progress SQL-89 Guide and Reference*

A user guide and reference for programmers who use interactive Progress/SQL-89. It includes information on all supported SQL-89 statements, SQL-89 Data Manipulation Language components, SQL-89 Data Definition Language components, and supported Progress functions.

### SQL-92

*Progress Embedded SQL-92 Guide and Reference*

A guide to Progress Embedded SQL-92 for C, including step-by-step instructions for building ESQL-92 applications and reference information about all Embedded SQL-92 Preprocessor statements and supporting function calls. This guide also describes the relationship between ESQL-92 and the ANSI standards upon which it is based.

*Progress JDBC Driver Guide*

A guide to the Java Database Connectivity (JDBC) interface and the Progress SQL-92 JDBC driver. It describes how to set up and use the driver and details the driver's support for the JDBC interface.

*Progress ODBC Driver Guide*

A guide to the ODBC interface and the Progress SQL-92 ODBC driver. It describes how to set up and use the driver and details the driver's support for the ODBC interface.

*Progress SQL-92 Guide and Reference*

A user guide and reference for programmers who use Progress SQL-92. It includes information on all supported SQL-92 statements, SQL-92 Data Manipulation Language components, SQL-92 Data Definition Language components, and Progress functions. The guide describes how to use the Progress SQL-92 Java classes and how to create and use Java stored procedures and triggers.

### Deployment

*Progress Client Deployment Guide*

A guide that describes the client deployment process and application administration concepts and procedures.

*Progress Developer's Toolkit*

A guide to using the Developer's Toolkit. This guide describes the advantages and disadvantages of different strategies for deploying Progress applications and explains how you can use the Toolkit to deploy applications with your selected strategy.

*Progress Portability Guide*

A guide that explains how to use the Progress toolset to build applications that are portable across all supported operating systems, user interfaces, and databases, following the Progress programming model.

### WebSpeed

*Getting Started with WebSpeed*

Provides an introduction to the WebSpeed Workshop tools for creating Web applications. It introduces you to all the components of the WebSpeed Workshop and takes you through the process of creating your own Intranet application.

*WebSpeed Installation and Configuration Guide*

Provides instructions for installing WebSpeed on Windows and UNIX systems. It also discusses designing WebSpeed environments, configuring WebSpeed Brokers, WebSpeed Agents, and the NameServer, and connecting to a variety of data sources.

*WebSpeed Developer's Guide*

Provides a complete overview of WebSpeed and the guidance necessary to develop and deploy WebSpeed applications on the Web.

*WebSpeed Product Update Bulletin*

A booklet that provides a brief description of each new feature of the release. The booklet also explains where to find more detailed information in the documentation set about each new feature.

*Welcome to WebSpeed!* (Hard copy only)

A booklet that explains how WebSpeed software and media are packaged. *Welcome to WebSpeed!* also provides descriptions of the various services Progress Software Corporation offers.

### Reference

*Pocket Progress* (Hard copy only)

A reference that lets you quickly look up information about the Progress language or programming environment.

*Pocket WebSpeed* (Hard copy only)

A reference that lets you quickly look up information about the SpeedScript language or the WebSpeed programming environment.

# 1

# Application Development Environment

The Progress Application Development Environment (ADE) consists of the tools you need to create a Progress application. This chapter describes the ADE in the following sections:

- Starting the ADE tools
- Accessing menus and menu options
- Navigating in dialog boxes and windows
- Using the Tools menu
- Using the Help menu

## 1.1 Starting the ADE Tools

The Progress Application Development Environment (ADE) is comprised of the following tools:

- Procedure Editor
- Data Dictionary
- Application Compiler

This manual covers the tools used for application development: Procedure Editor and Application Compiler. For information on the Data Dictionary, refer to the *Progress Basic Database Tools* manual. The following sections briefly describe each of the ADE tools and how to start them.

### 1.1.1 The Procedure Editor

The Procedure Editor lets you create and run Progress procedures. See Chapter 2, "Procedure Editor Tasks," for a description of the tasks you can perform with the Procedure Editor. See Chapter 4, "Procedure Editor Reference," for a description of the Procedure Editor menu options and dialog boxes.

Before starting the Procedure Editor, be sure to include the directory where you installed Progress in the PATH environment variable (`DLC\bin`, by default).

There are four ways to start the Progress Procedure Editor:

- To start a multi-user Progress session, enter the following command at the command line, where database refers to any database created with Progress Version 9:

```
mpro [ database ]
```

- To start a single-user Progress session, enter the following command at the command line, where database refers to any database created with Progress Version 9:

```
pro [ database ]
```

- To start a single-user Progress session and load procedure files, enter the following command at the command line:

```
pro -param
  " procedure1.p , procedure2.p , procedure3.p , ... "
```

- From any of the other ADE tools, choose Tools→ Procedure Editor.

### 1.1.2 The Application Compiler

The Application Compiler lets you compile a group of source procedures (.p and .w files). See Chapter 5, "Application Compiler," for a complete description of how to compile a Progress procedure.

To start the Application Compiler, choose Tools→ Application Compiler from the menu bar in any of the other Progress tools.

## 1.2 Accessing Menus and Menu Options

Follow these steps to access and execute the menu options in the ADE tools:

1 ♦ Press **F3** to access the menu bar for all tools except the Data Dictionary.

2 ♦ To choose a menu, do either of the following:

- Type the mnemonic (the underlined letter) in the menu name. For example, type **e** to choose the Edit menu.
- Use the arrow keys to highlight the menu name, then press **RETURN**.

Some menu options have predefined accelerator keys assigned to them, which let you use the keyboard to choose a menu option. *Accelerator keys* are function and special key combinations that let you use the keyboard to choose a menu option. If accelerator keys are assigned to an option, they appear to the right of the menu option on pull-down menus.

**NOTE:** In some cases, key combinations will not work if you use the **ESCAPE** key while the **CAPS LOCK** key is on.

For information about specifying keyboard mappings on your system, see the *Progress Client Deployment Guide*.

## 1.3 Navigating in Dialog Boxes and Windows

Many options you choose in the Progress tools display dialog boxes and windows where you have to enter information, toggle on boxes, or choose buttons. Use the following information as a guide to choosing options and entering information into dialog boxes and windows when using Progress tools:

- Move the highlight bar through the dialog box or window by pressing the right arrow or **TAB** key. For example, press **TAB** to skip a fill-in area, toggle box, or button.
- Reverse direction and move backward by pressing **BACKTAB**.
- Scroll up and down lists using the arrow keys.
- Choose a highlighted button or a toggle box by pressing **RETURN**.

## 1.4 Using the Tools Menu

The Tools menu lets you access a tool from another tool. For example, you can access the Data Dictionary from the Procedure Editor. However, access to other tools is only allowed in a forward direction. For example, you cannot return to the Procedure Editor from the Data Dictionary without closing the Dictionary. Table 1–1 lists the menu options that are available when you choose the Tools menu option.

**Table 1–1: Tools Menu**

| **Tools** | |
|---|---|
| Procedure Editor | Accesses the Procedure Editor. |
| Data Dictionary | Accesses the Data Dictionary. |
| OS Shell | Escapes to the operating system. |
| Application Compiler | Accesses the Application Compiler. |

### 1.4.1 Tools→ Procedure Editor

Choose this option to access the Progress Procedure Editor. The Procedure Editor lets you create, edit, compile, and run Progress procedures. See Chapter 2, "Procedure Editor Tasks," for a complete description of the tasks you can perform with the Procedure Editor.

### 1.4.2 Tools→ Data Dictionary

Choose this option to access the Progress Data Dictionary, which lets you create and modify database schema information including table, field, sequence, indexes, and trigger definitions. You can also generate schema reports. See the *Progress Basic Database Tools* manual for a complete description of the tasks you can perform with the Data Dictionary.

### 1.4.3 Tools→ OS Shell

Choose this option to temporarily leave the Procedure Editor so that you can run operating system commands from your operating system prompt. When you leave the shell, you automatically return to the Procedure Editor.

### 1.4.4 Tools→ Application Compiler

Choose this option to access the Application Compiler. The Application Compiler lets you compile a group of source procedures (.p files). See Chapter 5, "Application Compiler," for a complete description of the Application Compiler.

## 1.5 Using the Help Menu

All of the ADE tools, except the Data Dictionary, contain the Help menu. Help menu options provide access to online help information about the tool for which you are requesting help. Table 1–2 describes the menu that appears when you choose this option:

**Table 1–2: Help Menu**

| **Help** | |
|---|---|
| Messages... | Displays Progress messages. |
| Recent Messages... | Displays the most recent Progress message and a detailed description of the message. |
| Keyboard... | Accesses information about your keyboard and Progress keystrokes. |
| About *Tool*... | Shows information about this installation of the *Tool*. |

### 1.5.1 Help→ Messages

Choose this option to display a description of Progress messages. Enter a message number to display the description.

### 1.5.2 Help→ Recent Messages

Choose this option to display a description of the most recent message and all other messages generated by Progress during the current session.

### 1.5.3 Help→ Keyboard

Choose this option to show information about Editor and run-time key bindings. These lists show the actions associated with various keystrokes you can perform with your keyboard.

**NOTE:** In some cases, key combinations will not work if you use the **ESCAPE** key while the **CAPS LOCK** key is on.

### 1.5.4 Help→ About *Tool*

Choose this option to display information for a tool. It displays the version of Progress and the copyright date.

# 2

# Procedure Editor Tasks

This chapter describes many of the tasks you can perform using the Progress Procedure Editor, including:

- Starting the Procedure Editor
- Using edit buffers
- Working with procedures
- Editing text
- Compiling, running, checking, and debugging procedures
- Exiting the Procedure Editor

For details on the menu options and dialog boxes of the Procedure Editor, see Chapter 4, “Procedure Editor Reference.”

## 2.1 Starting the Procedure Editor

There are four ways to start the Progress Procedure Editor:

- From any of the other ADE tools, choose Tools→ Procedure Editor.
- To start a multi-user Progress session with the Procedure Editor, enter the following command at the command line, where *database* refers to any database created with Progress Version 9:

```
mpro [ database ]
```

- To start a single-user Progress session with the Procedure Editor, enter the following command at the command line, where *database* refers to any database created with Progress Version 9:

```
pro [ database ]
```

- To start a single-user Progress session and load procedure files, enter the following command at the command line:

```
pro -param
  " procedure1.p , procedure2.p , procedure3.p , ... "
```

When you start the Procedure Editor, the window shown in Figure 2–1 appears.

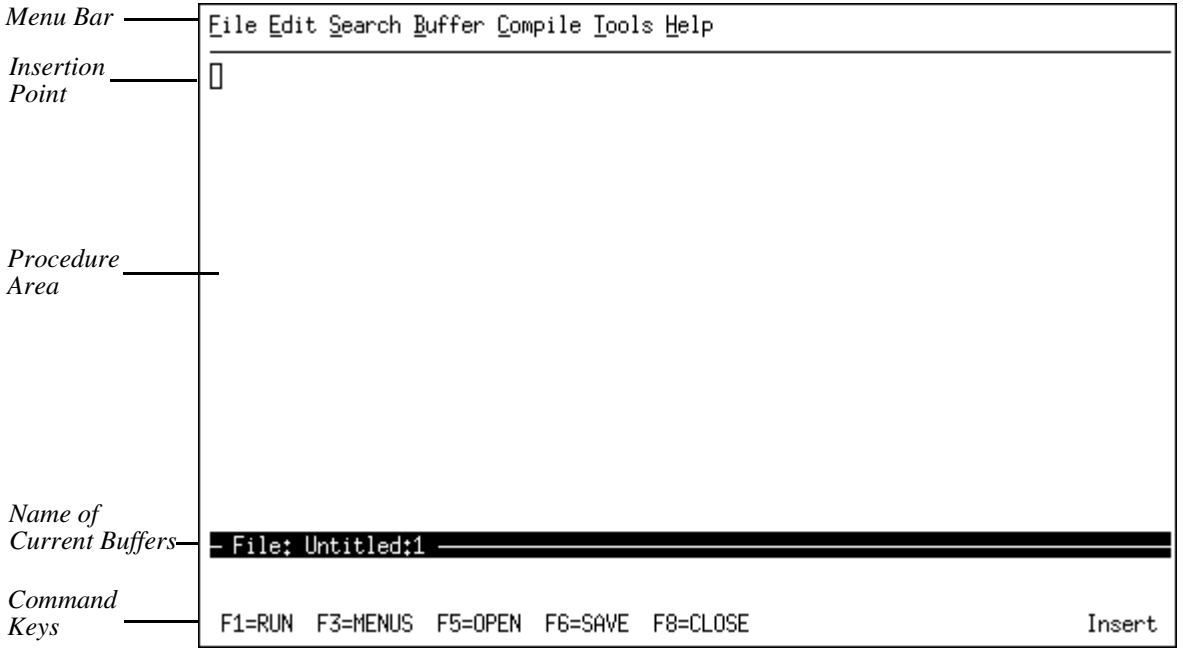



**Figure 2–1: Procedure Editor Window**

Table 2–1 describes the features of the Procedure Editor window:

**Table 2–1: Procedure Editor Features** *(1 of 2)*

| Feature | Purpose |
|---|---|
| Menu bar | Lets you access editor commands. |
| Insertion point | Marks where text appears when you start typing. |
| Procedure area | The visible part of the current buffer where you type and edit your Progress procedures. |

**Table 2–1: Procedure Editor Features** *(2 of 2)*

| Feature | Purpose |
|---|---|
| Current buffer name | The name of the procedure file you are currently editing. If the buffer has no name assigned, it appears as “Untitled” and is followed by a number to make it unique. |
| Command keys | Allow you to perform basic tasks in the Procedure Editor. These keys let you run procedures and access menus and help. They also let you open files into buffers and save or close them. |

**NOTE:** Dialog boxes, windows, or frames might appear in front of the window for various reasons. For example, they can display alert boxes that enable you to enter search strings and filenames or verify actions.

## 2.2 Using Edit Buffers

An *edit buffer* is a work area where you write and edit a Progress procedure. The Procedure Editor creates an edit buffer for each procedure you are working on, allowing you to open and edit multiple procedures in a single session. The number of buffers is limited by system memory.

**NOTE:** Although you can open multiple buffers in the Procedure Editor, you can open only one buffer for each operating system file.

The buffer displayed in the Procedure Editor’s procedure area is called the *current buffer*. When you enter text into the procedure area of the Procedure Editor, you enter it into the current buffer. When you start the Procedure Editor, the current buffer is empty and untitled unless you specify a procedure at the command line.

**NOTE:** The maximum buffer size you can open in the Procedure Editor is the lesser of the following:

- 32,767**bytesPerLine*, where *bytesPerLine* is the width of your terminal
- Available system resources

### 2.2.1 Listing Open Buffers

The Procedure Editor creates buffers when you use the New option or the Open option. To view a list of the open buffers for your current session, choose Buffer→ List. Progress lists the open buffers in the Buffer List dialog box in the order that you opened them, as shown in Figure 2–2. For more information on the Buffer List dialog box, see Chapter 4, "Procedure Editor Reference."

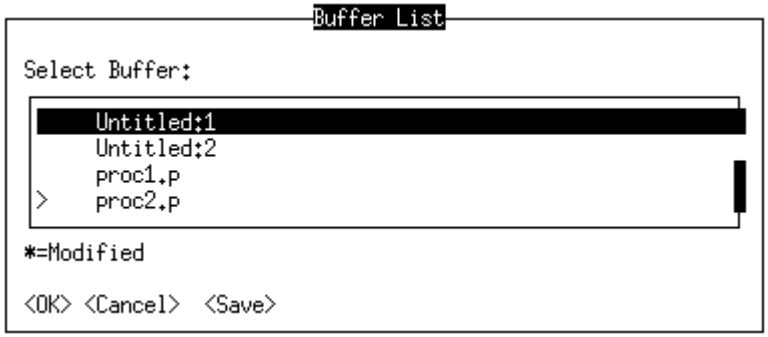



**Figure 2–2: Buffer List Dialog Box**

### 2.2.2 Switching Buffers

To switch to another open buffer, use one of the following techniques:

- Choose Buffer→ Next Buffer. The Procedure Editor makes the next buffer in the buffer list the current buffer.
- Choose Buffer→ Previous Buffer. The Procedure Editor makes the previous buffer in the buffer list the current buffer.
- Choose Buffer→ List, then select a buffer from the Buffer List dialog box and choose OK. The Procedure Editor displays the buffer you selected as the current buffer.

### 2.2.3 Displaying Buffer Information

To display information about the current buffer, choose Buffer→ Information. Progress displays the Buffer Information dialog box, which contains the filename; read and write access for the file; the number of lines, columns, and bytes; and the modified status of the buffer. Choose OK to close the dialog box. See Chapter 4, "Procedure Editor Reference," for more information on the Buffer Information dialog box.

## 2.3 Working with Procedures

This section describes how to manipulate procedures in the Procedure Editor. It describes the following tasks:

- Creating a new procedure
- Opening a procedure
- Saving a procedure
- Printing a procedure
- Closing a procedure

### 2.3.1 Creating a New Procedure

To create a new procedure, choose File→ New from the menu bar. Progress creates a new buffer in the Procedure Editor and makes the new buffer the current buffer. The Procedure Editor adds the new buffer to the buffer list, labels it "Untitled," and gives it a number. To use this new buffer to write a Progress procedure, enter code and choose the Save As option to name and store the procedure.

**NOTE:** The Procedure Editor does not allow you to create a file with a comma in its name.

### 2.3.2 Opening a Procedure

Follow these steps to open an existing procedure file:

1 ♦ Choose File→ Open.

2 ♦ Select the filename from the Open dialog box, then choose OK.

**NOTE:** The Procedure Editor cannot open a file with a comma in its name.

The Procedure Editor creates a new buffer, reads the file into it, makes it the current buffer, and labels it with the name of the file. You can open multiple buffers in the Procedure Editor, but you cannot open more than one buffer at a time for the same operating system file. The number of buffers is limited by system memory. You can access other buffers from the Buffer menu. For more information on the Buffer menu, see Chapter 4, "Procedure Editor Reference."

**NOTE:** The maximum buffer size you can open in the Procedure Editor is the lesser of the following:

- 32,767**bytesPerLine*, where *bytesPerLine* is the width of your terminal
- Available system resources

## 2.3.3 Saving a Procedure

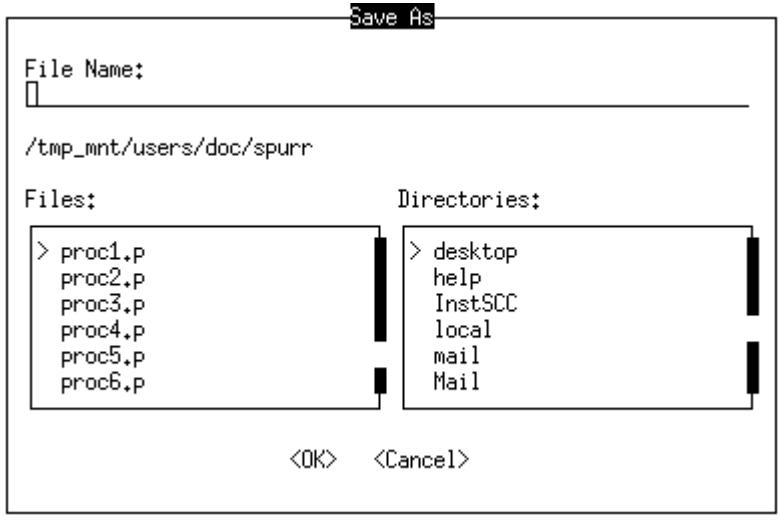
You can save the contents of the current buffer in the following ways:

- By saving a procedure to a new file
- By saving a procedure to the current file

### Saving a Procedure to a New File

Follow these steps to save a new, untitled procedure to a file:

1 ♦ Choose File→ Save or choose File→ Save As from the menu bar. The Save As dialog box appears. See Chapter 4, "Procedure Editor Reference," for a detailed description of the Save As dialog box fields:




2 ♦ Specify the file where you want to save the procedure.

When you enter a filename, add the .p extension to the procedure file. Procedures are stored as operating system files. UNIX filenames are case sensitive, unlike Windows.

Unless you specify a path when you name the file, Progress stores the file in your current working directory.

**CAUTION:** Do not name a procedure `_edit.p`, because that is the name of the Procedure Editor. If you do, and the procedure falls before $DLC in your PROPATH, Progress accesses the incorrect file when you try to run the Procedure Editor.

Progress saves the procedure to the specified filename and keeps the buffer open so you can continue working in it. The buffer name changes to the specified filename.

**NOTE:** The Procedure Editor cannot save a file with a comma in its name.

#### Saving a Procedure to the Current File

When you are in a named buffer and you choose File→ Save, Progress automatically saves the procedure to the current filename and keeps the buffer open so you can continue working in it.

### 2.3.4 Printing a Procedure

To print the contents of the current buffer, choose File→ Print. Progress sends the current buffer contents to your computer system's default printer. There are no print format options.

### 2.3.5 Closing a Procedure

To close a procedure, choose File→ Close. The Procedure Editor closes the current buffer. If you make changes to the current buffer, Progress prompts you to save the changes before it deletes the buffer. If the buffer is untitled and you choose to save it, Progress displays the Save As dialog box. Specify the filename and choose OK to save the procedure.

## 2.4 Editing Text

This section describes how to edit text in the Procedure Editor. It describes the following tasks:

- Entering text
- Selecting text
- Cutting, copying, and pasting text
- Searching for text
- Inserting a file into a procedure
- Inserting field names into a procedure

### 2.4.1 Entering Text

Enter text into the Procedure Editor by typing as you would in any other editor. Text you enter into the current buffer appears at the location of the cursor. The *cursor* is a place marker in text. The location of the cursor is referred to as the *insertion point*.

You can use the keyboard arrow keys to relocate the insertion point. You can also choose Search→ Goto Line to move the insertion point to a specific line number within the current buffer. See the "Search→ Goto Line" section in Chapter 4, "Procedure Editor Reference" for more information on this option.

#### Entering Special Characters

There are some characters that you cannot type directly into the edit buffer. To represent ASCII control characters or other character codes that the keyboard cannot generate directly (8-bit codes, for example), type the three-digit octal code of the character, preceded by a tilde (~). For more information about special characters, see the *Progress Language Reference*.

When you write procedures with the Procedure Editor, you can use uppercase, lowercase, or mixed case. The Progress Compiler recognizes that table, TABLE, and Table are the same word.

#### Entering Long Text Lines

You can create and edit text lines longer than 80 characters using the Procedure Editor.

You can use a tilde (~) as the last non-blank character on a line to indicate that the line is continued, beginning with column 1 of the next line in the window, as shown in the following example:

```
DISPLAY "This is a long message ~
continued on the next line".
```

The Progress Compiler interprets the lines in this example as the following single line:

```
DISPLAY "This is a long message continued on the next line".
```

The Procedure Editor does not change the physical presentation you give the file. When you use a tilde to connect two lines, the Procedure Editor maintains the tilde in the code. The Procedure Editor saves it and displays it. However, the Compiler compiles the file without the tilde.

### 2.4.2 Selecting Text

Select text when you want to perform an action or command on that text or if you want to use the selected text in a command.

You can select text by positioning the cursor where you want to begin selecting text and pressing **CTRL-V** to enter block selection mode. Then move the cursor to the end of the text you want to select. The text is not highlighted, but it is selected.

### 2.4.3 Cutting, Copying, and Pasting Text

Follow these steps to cut or copy text in a procedure:

1 ♦ Select the text you want to cut or copy. See the "Selecting Text" section for the steps to follow.

2 ♦ To cut the text, choose Edit→ Cut. To copy the text, choose Edit→ Copy.

The Procedure Editor deletes or copies the selected range of text from the file in the current buffer and places it onto the clipboard.

Follow these steps to paste text in a procedure:

1 ♦ Position the cursor at the point you want to insert the text.

2 ♦ Choose Edit→ Paste.

The Procedure Editor inserts the contents of the clipboard at the cursor and repositions the cursor after the pasted text.

### 2.4.4 Searching for Text

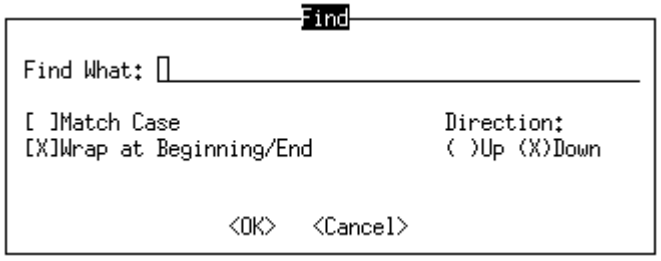
You can search for text in the current buffer using the following menu options:

- Finding text
- Finding the next or previous occurrence of text
- Replacing text

### Finding Text

Follow these steps to search for text in the current buffer:

1 ♦ Choose Search→ Find. Progress displays the Find dialog box:




2 ♦ Enter the text string you want to find.

If you have searched for a string in the current session, the Find What field displays the text string for which you last searched. If your search is not case sensitive (that is, you do not select the Match Case option in the Find dialog box), the string you supply can be uppercase, lowercase, or both. Otherwise, enter it exactly as you want to search for it.

3 ♦ Specify whether the search is case sensitive using the Match Case option.

4 ♦ Specify whether to wrap to the beginning or end of the current buffer when the search reaches the opposite end of the buffer.

5 ♦ Specify whether to search forward or backward through the current buffer by selecting the up or down options.

6 ♦ Choose OK.

The Procedure Editor searches for the first occurrence of the string in the direction you specify. When it finds a match for the search string, it positions the cursor at the end of the search string. You can choose Search→ Find Next or Search→ Find Previous to find the next or previous occurrence of the text. The Procedure Editor displays an alert box if it does not find a match.

### Finding the Next or Previous Occurrence of Text

If you have used the Search→ Find option in the current session, you can search for text using the Search→ Find Next or Search→ Find Previous menu options. The Procedure Editor searches for the text you specified in the Find What field of the Find dialog box in the forward direction (Next) or reverse direction (Previous).

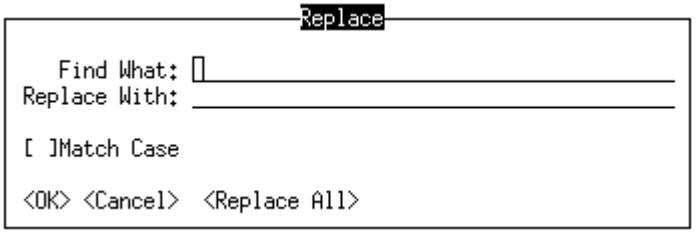
If you have not previously used the Search→ Find option in this session, Progress displays an alert box when you select Search→ Find Next or Search→ Find Previous.

### Replacing Text

Follow these steps to search for and replace text in the current buffer:

1 ♦ Choose Search→ Replace.

2 ♦ Enter the text string you want to find. The Replace dialog box appears:




If you have searched for a string in the current session, the Find What field displays the string for which you last searched. If your search is not case sensitive (that is, you do not select the Match Case option in the Replace or Find dialog box), the string you supply can be uppercase, lowercase, or both. Otherwise, enter it exactly as you want to search for it. You cannot use wildcard characters when specifying the find string.

3 ♦ Enter the string with which you want to replace the specified string.

4 ♦ Specify if the search is case sensitive.

5 ♦ Choose the Replace All button if you want to replace all occurrences of the search string with the new string without confirming each occurrence.

6 ♦ Choose OK.

The Procedure Editor searches for the first occurrence of the string in a forward direction. If you do not choose the Replace All button, the Editor prompts you to confirm the replacement for each occurrence of the string it finds. If you choose the Replace All button, the Editor automatically replaces each occurrence of the string with the text in the Replace With field. When complete, the Editor displays an information alert box indicating the Replace All task is complete and displaying the number of occurrences replaced.

### 2.4.5 Inserting a File into a Procedure

Follow these steps to insert the entire contents of a file into the text in the current buffer:

1 ♦ Place the cursor at the point where you want to insert the file.

2 ♦ Choose Edit→ Insert File from the main menu. The Insert File dialog box appears.

3 ♦ Choose the file you want to insert, then choose OK.

The Procedure Editor inserts the file contents into the current buffer at the cursor position.

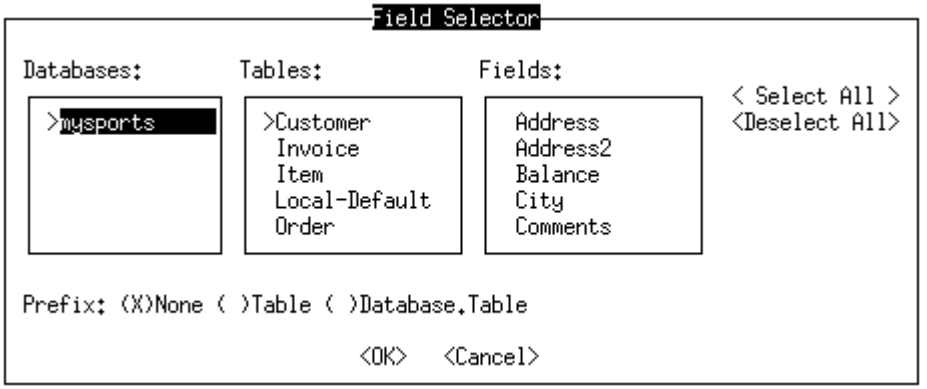
### 2.4.6 Inserting a Field Name into a Procedure

Follow these steps to insert a field name into a procedure in the current buffer:

1 ♦ Place the cursor at the point where you want to insert the field name.

2 ♦ Choose Edit→ Insert Fields from the main menu. If you are connected to a database, Progress displays the Field Selector dialog box, as shown below.

If you are not already connected to a database, Progress displays an alert box. If you choose OK, Progress displays the Database Connect dialog box. See the *Progress Basic Database Tools* manual for information on this dialog box. After you connect to a database, Progress displays the Field Selector dialog box:




3 ♦ Choose the fields you want to insert.

4 ♦ Specify prefixes to include with the field names. See the "Edit→ Insert Fields" section in Chapter 4, "Procedure Editor Reference," for more information on these options.

5 ♦ Choose OK. The Procedure Editor inserts the field names into the current buffer at the cursor position.

## 2.5 Compiling, Running, and Checking Procedures

The Procedure Editor supports the full edit-compile-run cycle. This support includes:

- Checking a procedure's syntax
- Compiling and running a procedure
- Debugging a procedure

### 2.5.1 Checking a Procedure's Syntax

To check a procedure's syntax, choose Compile→ Check Syntax from the menu bar. The Procedure Editor checks the procedure's syntax and displays all applicable messages in the Compiler Messages dialog box.

### 2.5.2 Compiling and Running a Procedure

To compile and run a procedure in the current buffer, choose Compile→ Run from the menu bar. The Procedure Editor accesses the Progress 4GL Compiler, which compiles the procedure. If the procedure does not compile, Progress displays all applicable messages in the Compiler Messages dialog box.

The messages displayed typically show a brief description of the message, often including the name of the file and line number containing the error, followed by a Progress message number in parentheses. You can display additional information about the messages using the Help→ Messages or Help→ Recent Messages options.

If the Progress 4GL Compiler detects any errors, the Procedure Editor moves the text cursor to the line in the current buffer that contains the first error. If the file containing the error is not in the current buffer but is an include file, the Procedure Editor opens the include file, making it the current buffer. The Procedure Editor then positions the cursor on the line in the include file that contains the error. If the include file containing the error is not syntactically complete (that is, it does not contain at least one 4GL statement with a period), the Procedure Editor opens the source file that references the include statement, rather than the include file that contains the error.

When you run a procedure and have several open buffers, the Progress 4GL Compiler accesses only the code in the current buffer of the Procedure Editor. It does not access files in other buffers but uses the saved versions of the files. This means that if you run a procedure, and the Progress 4GL Compiler detects an error in a file open in a buffer other than the current buffer, you must save any changes you make to that called file in order for them to be recognized when you rerun the initial procedure.

If you do not save the changes, the next time you run the initial procedure that calls the file, the Progress 4GL Compiler runs the saved version of the files and disregards the changes you have made in the open buffer of the Procedure Editor. When the Progress 4GL Compiler detects the error, the Procedure Editor switches the called file to the current buffer and places the cursor on the line that previously contained the error. Your changes appear in the current buffer, but because you did not save them, the Progress 4GL Compiler could not access the modified code.

### 2.5.3 Debugging a Procedure

To debug a procedure, choose Compile→ Debug from the menu bar. The Procedure Editor checks the procedure's syntax. If the procedure compiles, the Procedure Editor opens the Progress Debugger and displays the procedure at the break point of the first executing line. For more information about debugging a procedure, see the *Progress Debugger Guide*.

**NOTE:** The Debugger requires a Motif environment.

## 2.6 Exiting the Procedure Editor

You can exit the Procedure Editor in two ways:

- Choose File→ Exit.
- Type **quit** in an empty buffer, then choose Compile→ Run or press **GO**.

If there are open buffers with unsaved changes, an alert box prompts you to save or discard the buffers before leaving the Procedure Editor. To discard the buffers and exit the Procedure Editor, choose No in the alert box. Follow these steps to save any of the changed buffers:

1 ♦ Choose Yes in the alert box. Progress displays the Save Buffers with Changes dialog box.

2 ♦ Select the buffers to save and choose the Save Selected button. If you are saving any untitled buffers, the Save As dialog box appears for each buffer.

3 ♦ Specify the filename for each untitled buffer, then choose OK. After you specify the filename for the last untitled buffer, Progress closes the Procedure Editor.

When Progress closes the Procedure Editor, it returns to where you started. For example, if you start the Procedure Editor from the operating system command line, Progress returns to the command line.

# 3

# Procedure Editor Integration Hooks

This chapter explains how to add integration hooks to the Procedure Editor by describing the parameters and events for `adecomm/_adeevnt.p` and how to modify this procedure to interface with SCM or your own proprietary tools:

- ADE Event (`adecomm/_adeevnt.p`)
- Parameters
- Events
- Usage

Source code management (SCM) tools or your own proprietary tools can interface with the Procedure Editor. They can intercept and augment tool behavior at the following critical points in the application development process:

- Opening files
- Saving files
- Closing files
- Before and after a file is run, debugged, or checked for syntax
- Startup and shutdown of the Procedure Editor

To allow SCM or your own proprietary tools to "trap" these ADE events, the Procedure Editor calls a procedure file—`adecomm/_adeevnt.p`—at those critical processing points. You can modify this procedure to intercept and augment standard tool behavior. The source code to this procedure is provided by Progress Software. If you want to use this procedure, you must copy the source code, modify it, compile it, and place it in your PROPATH.

**NOTE:** For GUI application development, there are three procedure files for integration hooks—`adecomm/_adeevnt.p`, `adecomm/_getfile.p`, and `adecomm/_chosobj.w`. For more information on how to use integration hooks for GUI application development, see the online help.

## 3.1 ADE Event (adecomm/_adeevnt.p)

You can modify `adecomm/_adeevnt.p` to intercept or filter various events that occur in an ADE session. The Procedure Editor calls `adecomm/_adeevnt.p` when certain processing events occur. Generally, these events are file oriented—for example, OPEN, SAVE, CLOSE—but they can be more general—for example, STARTUP and SHUTDOWN.

The source code for this file is in `DLC/src/adecomm/_adeevent.p`. Look at this file for the latest information on ADE events.

## 3.2 Parameters

This section describes the input and output parameters for the `adecomm/_adeevnt.p` procedure.

### 3.2.1 Input Parameters

The `adecomm/_adeevnt.p` procedure takes the following input parameters:

p_product

The ADE product code (for the Procedure Editor, use "Editor").

p_event

The name of an event: NEW, OPEN, BEFORE-OPEN, CLOSE, BEFORE-CLOSE, SAVE, BEFORE-SAVE, COMPILE, BEFORE-COMPILE, RUN, BEFORE-RUN, DEBUG, BEFORE-DEBUG, CHECK-SYNTAX, BEFORE-CHECK-SYNTAX, CHECK-SYNTAX-PARTIAL, and BEFORE-CHECK-SYNTAX-PARTIAL.

p_context

A string that is used to uniquely identify the file being edited. The Procedure Editor uses the edit widget-handle of the buffer for this value. A specific file will have the same context ID for all its file operations. However, if you close a file and then open it again, the context number will change.

p_other

Additional information passed about an event (for example, a "SAVE" event normally passes the filename for the save).

The current filename associated with the window. The name will be unknown (?) if it has not been set—for example, after a File→ New.

### 3.2.2 Output Parameters

The `adecomm/_adeevnt.p` procedure uses the following output parameters:

p_ok

A logical value used to OK or cancel a subset of ADE events. For example, returning a FALSE value from the "BEFORE-SAVE" event cancels the save.

## 3.3 Events

This section describes the file-specific events and the START-UP and SHUTDOWN events for the `adecomm/_adeevnt.p` procedure.

### 3.3.1 File-specific Events

Table 3–1 describes the events (corresponding to p_event) that are related to file operations:

**Table 3–1: File-specific Events** *(1 of 2)*

| Event | Operation |
|---|---|
| NEW | Called after a new window/dialog box is created. |
| OPEN | Called after a file has been opened. |
| BEFORE-OPEN | Called before a file is to be opened; returning p_ok as FALSE cancels the operation. |
| CLOSE | Called after a window or buffer has been closed. |
| BEFORE-CLOSE | Called before a file is to be closed; returning p_ok as FALSE cancels the operation. |
| SAVE | Called after a file has been saved. |
| BEFORE-SAVE | Called before a file is to be saved; returning p_ok as FALSE cancels the operation. |
| COMPILE | Called after a file has been compiled. |

**Table 3–1: File-specific Events** *(2 of 2)*

| Event | Operation |
|---|---|
| BEFORE-COMPILE | Called before a file is to be compiled; returning p_ok as FALSE cancels the operation. |
| RUN | Called after RUN of file has ended. |
| BEFORE-RUN | Called before a file has been written to disk for a run; returning p_ok as FALSE cancels the operation. |
| DEBUG | Same as RUN, except that DEBUG has been chosen; returning p_ok as FALSE cancels the operation. |
| BEFORE-DEBUG | Same as BEFORE-RUN, except DEBUG is chosen. |
| CHECK-SYNTAX | Called after a Check Syntax. |
| BEFORE-CHECK-SYNTAX | Called before a Check Syntax; returning p_ok as FALSE cancels the operation. |
| CHECK-SYNTAX-PARTIAL | Called after a Partial Check Syntax. |
| BEFORE-CHECK-SYNTAX-PARTIAL | Called before a Partial Check Syntax; returning p_ok as FALSE cancels the operation. |

**NOTE:** Some events, for example NEW, cannot be cancelled even if returning FALSE.

### 3.3.2 STARTUP and SHUTDOWN Events

The following Procedure Editor events (corresponding to p_event) occur at Procedure Editor startup and shutdown:

| Event | Operation |
|---|---|
| STARTUP | Called when the Procedure Editor (PE) has been loaded and initialized. This call occurs immediately before user input is allowed. In this case:<br>• p_context = STRING(*procedure-handle-of-the-PE-main-routine*).<br>• p_other = STRING(*widget-handle-of-the-PE-window*). |
| SHUTDOWN | Called when a user requests that the Procedure Editor shutdown. This call occurs before any settings have been saved or items destroyed. In this case:<br>• p_context = STRING(*procedure-handle-of-the-PE-main-routine*).<br>• p_other - STRING(*widget-handle-of-the-PE-window*). |

**NOTE:** The Procedure Editor STARTUP and SHUTDOWN events cannot be cancelled.

## 3.4 Usage

The following comments address Procedure Editor usage issues:

- BEFORE-CLOSE and CLOSE—technically, p_other should be UNKNOWN (?) after a file closes; however, this parameter still shows the last available file name for the procedure file. Unknown (?) only appears if there is no file name.
- BEFORE-RUN, RUN, BEFORE-DEBUG, DEBUG, BEFORE-CHECK-SYNTAX, CHECK-SYNTAX—all use the last specified file name as p_other. The actual file being run or checked is a temporary file with a name like `p01928384.ped`. This name is not used.
- If the user tries to close a buffer, then the Procedure Editor first prompts the user to save. The entire save operation events fire before the call to BEFORE-CLOSE.
- NEW is called after a buffer is created. You see the buffer before the NEW event is called. All events are called after the event has finished.
- When you control the handle for the Procedure Editor window, you can manipulate the display of the window. For example, you can add a menu option.

The scenario shown in Table 3–2 applies to the Procedure Editor.

**Table 3–2: Usage Scenario for the Procedure Editor** *(1 of 2)*

| Action | p_event | p_context | p_other | Comments |
|---|---|---|---|---|
| **Creating, Saving, and Running the File** | | | | |
| File→ New | NEW | 56788 | ? | File name unknown. |
| File→ Save | BEFORE-SAVE | 56788 | `c:\9\window-1.w` | – |
| | SAVE | 56788 | `c:\9\window-1.w` | – |
| Running the file | BEFORE-RUN | 56788 | `c:\9\window-1.w` | Run uses last file name. |
| | RUN | 56788 | `c:\9\window-1.w` | – |
| File→ Save As | BEFORE-SAVE | 56788 | `c:\9\my-file.w` | Same context, new name. |
| | SAVE | 56788 | `c:\9\my-file.w` | – |
| **Closing the File with Modifications** | | | | |
| Procedure Editor prompts user to save changes | BEFORE-SAVE | 56788 | `c:\9\my-file.w` | Closing a file can cause the Procedure Editor to prompt the user to save a file. |
| | SAVE | 56788 | `c:\9\my-file.w` | – |
| Procedure Editor closes the file | BEFORE-CLOSE | 56788 | `c:\9\my-file.w` | – |
| | CLOSE | 56788 | `c:\9\my-file.w` | – |
| **Opening the File** | | | | |
| File→ Open | BEFORE-OPEN | ? | `c:\9\my-file.w` | Occurs after the open dialog box. Before-open can be cancelled. |
| | OPEN | 46647 | `c:\9\my-file.w` | Observe the new context number. |

**Table 3–2: Usage Scenario for the Procedure Editor** *(2 of 2)*

| Action | p_event | p_context | p_other | Comments |
|---|---|---|---|---|
| **Closing the File Without Modifications** | | | | |
| File→ Close | BEFORE-CLOSE | 46647 | `c:\9\my-file.w` | – |
| | CLOSE | – | – | – |

# 4

# Procedure Editor Reference

This chapter describes the Procedure Editor's menu options and dialog boxes. The Procedure Editor is a menu-driven tool you can use to create, edit, compile, and run Progress procedures. The Procedure Editor is fully integrated with the Progress 4GL Compiler, the Progress Data Dictionary, and the Progress Debugger. It checks the syntax of your procedures and verifies the existence of tables and fields named in your Progress procedures.

For information on how to use the Procedure Editor to perform tasks, see Chapter 2, "Procedure Editor Tasks."

## 4.1 Procedure Editor Menu Bar

The following sections describe the menus and options available from the Procedure Editor. Figure 4–1 shows the Procedure Editor menu bar. To access the menu bar, press **F3**.

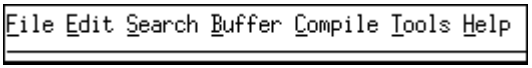



**Figure 4–1: Procedure Editor Menu Bar**

Table 4–1 describes the Procedure Editor menus.

**Table 4–1: Procedure Editor Menus**

| Menu | Description |
|---|---|
| File | Manages files and exits the Procedure Editor. |
| Edit | Manipulates and edits blocks of code. |
| Search | Searches for and replaces code strings in the current buffer. |
| Buffer | Manages buffers, including switching the current buffer and displaying buffer information. |
| Compile | Runs and compiles Progress procedure files. |
| Tools | Accesses other Progress tools. |
| Help | Accesses information about Progress error messages and keyboard mappings. |

## 4.2 File Menu

File menu options enable you to retrieve and save Progress program files. Table 4–2 describes the menu that appears when you choose this option.

**Table 4–2: File Menu** *(1 of 2)*

| **File** | |
|---|---|
| New | Creates a new file or procedure. |
| Open... | Opens an existing file or procedure to edit. |

**Table 4–2: File Menu** *(2 of 2)*

| File | |
|---|---|
| Close | Closes the current buffer. |
| Save | Saves the contents of the current buffer. |
| Save As... | Saves the contents of the current buffer to a file. |
| Print | Prints the contents of the current buffer. |
| Exit | Ends your current Procedure Editor session. |

### 4.2.1 File→ New

Choose this option to create a new file or procedure. The Procedure Editor creates a new, untitled buffer and makes it the current buffer. The number of buffers is limited by system memory.

**NOTE:** The maximum buffer size you can open in the Procedure Editor is the lesser of the following:

- 32,767**bytesPerLine*, where *bytesPerLine* is the width of your terminal
- Available system resources

You can switch between buffers using the Buffer menu commands. For more information, see the “Buffer Menu” section later in this chapter.

### 4.2.2 File→ Open

Choose this option to edit an existing file. When you choose this option, the Open dialog box shown in Figure 4–2 appears.

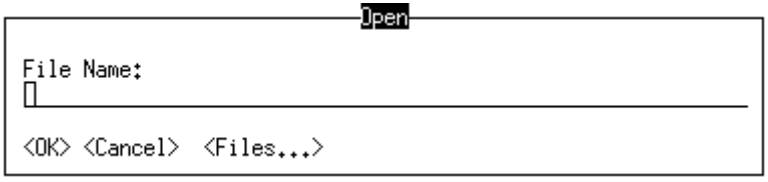



**Figure 4–2: Open Dialog Box**

**NOTE:** The Procedure Editor cannot open a file with a comma in its name.

When you choose the Files button from the Open dialog box, the Files dialog box similar to the one shown in Figure 4–3 appears.

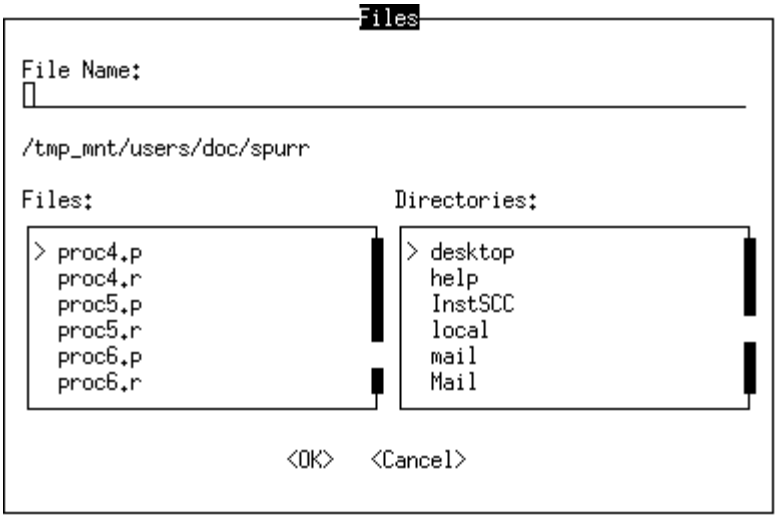



**Figure 4–3: Files Dialog Box**

Table 4–3 describes the user-interface elements of the Files dialog box.

**Table 4–3: Files Dialog Box**

| User-interface Element | Purpose |
|---|---|
| File Name | Specifies the name of the file you want to open. |
| Files | Lists the files in the currently selected directory. |
| Directories | Shows the currently selected directory. The directory selection list displays all the available directories. |

### 4.2.3 File→ Close

Choose File→ Close to delete the current buffer. If you make changes to the current buffer, the Procedure Editor prompts you to save the changes to a file before it deletes the buffer. If the buffer is untitled, an alert box similar to the one shown in Figure 4–4 appears.

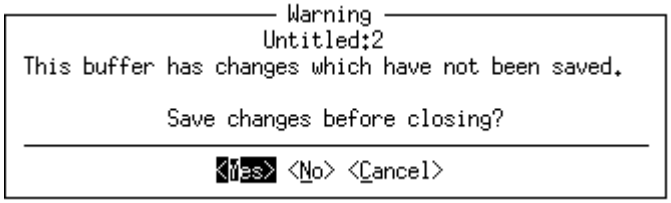



**Figure 4–4: Close Alert Box**

### 4.2.4 File→ Save

Choose this option to save the contents of the current buffer to the current file. When you choose this option and the current buffer is untitled, Progress displays the Save As dialog box shown in Figure 4–5. The Procedure Editor displays an alert box if you try to save changes to a read-only file.

### 4.2.5 File→ Save As

Choose File→ Save As to save the contents of the current buffer to a new file or to an existing file. When you choose this option, the Save As dialog box shown in Figure 4–5 appears.

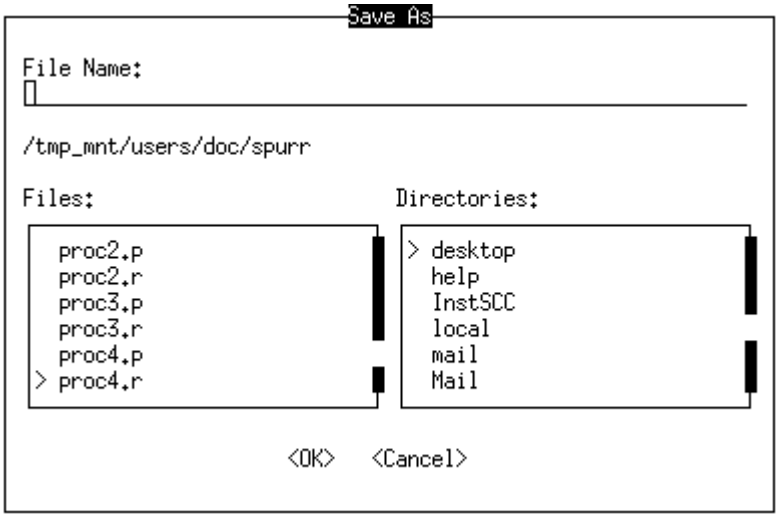



**Figure 4–5: Save As Dialog Box**

**NOTE:** The Procedure Editor cannot save a file with a comma in its name.

Table 4–4 describes the fields in the Save As dialog box.

**Table 4–4: Save As Dialog Box Fields**

| Fields | Usage |
|---|---|
| File Name | Specifies the name for the file you want to save. |
| Files | Lists the files in the selected directory. |
| Directories | Lists the directories in your file system. |

### 4.2.6 File→ Print

Choose this option to print the contents of the current buffer. The current buffer contents are sent to your computer system's default printer. There are no special formatting features or options.

### 4.2.7 File→ Exit

Choose this option to close the Procedure Editor. When you exit the Procedure Editor, you return to where you started. For example, if you start the Procedure Editor from the operating system command line, you return to the command line.

If there are open buffers with unsaved changes, Progress displays an alert box that prompts you to save or discard the changes before leaving the Procedure Editor. If you choose to save the changes, the Save Buffers with Changes dialog box appears as shown in Figure 4–6. This dialog box lists all the open buffers that have unsaved changes.

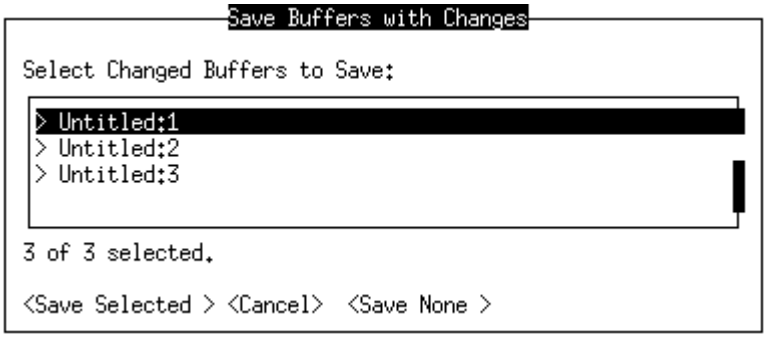



**Figure 4–6: Save Buffers with Changes Dialog Box**

If you select any untitled buffers and choose the Save Selected button, the Save As dialog box appears for each untitled buffer. You can specify a filename for each untitled buffer or cancel the save operation.

## 4.3 Edit Menu

Edit menu options enable you to manipulate and edit blocks of text. Table 4–5 describes the menu that appears when you choose this option.

**Table 4–5: Edit Menu** *(1 of 2)*

| **Edit** | |
|---|---|
| Cut | Removes selected text from the current buffer and stores it on the system clipboard. |
| Copy | Copies selected text onto the system clipboard. |
| Paste | Places clipboard contents in the current buffer at the insertion point. |

**Table 4–5: Edit Menu** *(2 of 2)*

| **Edit** | |
|---|---|
| Insert File... | Copies the contents of the selected file into the current buffer at the insertion point. |
| Insert Fields... | Inserts selected database field names into the current buffer at the insertion point. |

### 4.3.1 Edit→ Cut

Choose this option to delete the selected text from the file in the current buffer and place it onto the system clipboard, erasing the previous clipboard contents.

Choose the Edit→ Paste option to retrieve the deleted text from the clipboard and insert it into the current buffer.

**NOTE:** To select text, position the cursor where you want to begin selecting text and press **CTRL-V** to enter block selection mode. Then move the cursor to the end of the text you want to select. The text is not highlighted, but it is selected.

### 4.3.2 Edit→ Copy

Choose this option to make a copy of the selected text from the file in the current buffer and place it onto the system clipboard, erasing the previous clipboard contents.

Choose the Edit→ Paste option to retrieve the copied text from the clipboard and insert it into the current buffer.

**NOTE:** To select text, position the cursor where you want to begin selecting text and press **CTRL-V** to enter block selection mode. Then move the cursor to the end of the text you want to select. The text is not highlighted, but it is selected.

### 4.3.3 Edit→ Paste

Choose this option to place the contents of the system clipboard at the insertion point in the file in the current buffer. If text is selected, the Paste option replaces it with the contents of the system clipboard.

### 4.3.4 Edit→ Insert File

Choose Edit→ Insert File to read a copy of a file into the current buffer. When you choose this option, the Insert File dialog box shown in Figure 4–7 appears.

The Edit→ Insert File option works the same way as the File→ Open option, except that it pulls the file information into the current buffer instead of a new buffer.

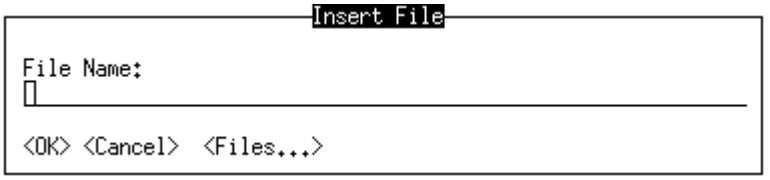



**Figure 4–7: Insert File Dialog Box**

## 4.3.5 Edit→ Insert Fields

Choose Edit→ Insert Fields to insert a field name into the current buffer. When you choose Insert Fields, the Field Selector dialog box shown in Figure 4–8 appears.

**NOTE:** If you are not already connected to a database, Progress displays an alert box and lets you access the Database Connect dialog box.

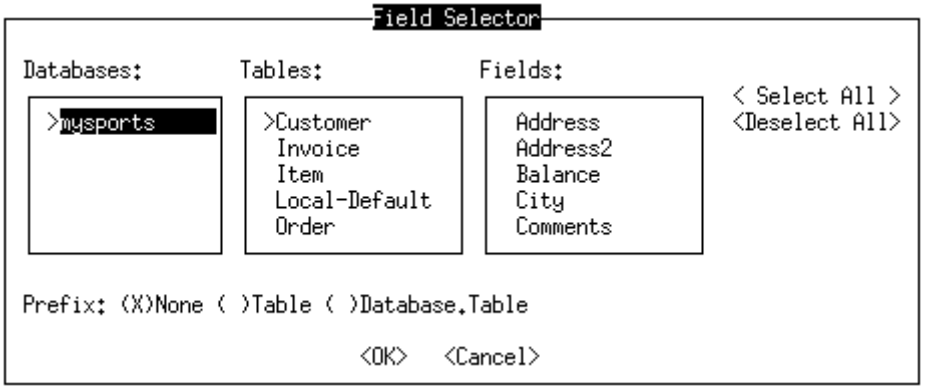



**Figure 4–8: Field Selector Dialog Box**

Table 4–6 describes the user-interface elements of the Field Selector dialog box.

**Table 4–6: Field Selector Dialog Box**

| User-interface Element | Purpose |
|---|---|
| None | Indicates that you do not want to include a prefix with the field name in your procedure. |
| Table | Indicates that you want to include the table name as well as the field name in your procedure. |
| Database.Table | Indicates that you want to include the database and table name with the field name in your procedure. |

## 4.4 Search Menu

Search menu options enable you to search for and replace text strings in the current buffer. Table 4–7 describes the menu that appears when you choose this option.

**Table 4–7: Search Menu**

| **Search** | |
|---|---|
| Find... | Searches for specific text. |
| Find Next | Searches for the next occurrence of the text string specified in the Find dialog box. |
| Find Previous | Searches for the previous occurrence of the text string specified in the Find dialog box. |
| Replace... | Searches for and changes specified text. |
| Goto Line... | Moves the insertion point to a specified line number in the current buffer. |

### 4.4.1 Search→ Find

Choose Search→ Find to search for text within a file. When you choose Search→ Find, the Find dialog box shown in Figure 4–9 appears.

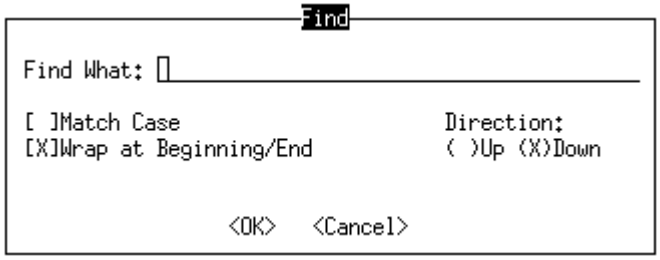



**Figure 4–9: Find Dialog Box**

Table 4–8 describes the fields in the Find dialog box.

**Table 4–8: Find Dialog Box**

| Field | Purpose |
|---|---|
| Find What | Identifies the text to search for. |
| Match Case | Specifies whether the search is case sensitive. |
| Wrap at Beginning/End | Specifies whether to wrap to the beginning of the current buffer and continue searching when the search reaches the end of the buffer. |
| Direction | Specifies whether to search forward or backward through the current buffer. Choose the appropriate radio button to indicate the search direction. |

### 4.4.2 Search→ Find Next

Choose this option to find the next occurrence of the text for which you most recently searched using the Search→ Find option. This option uses the same search criteria as the Search→ Find option.

### 4.4.3 Search→ Find Previous

Choose this option to find the previous occurrence of the text for which you most recently searched using the Search→ Find option. This option uses the same search criteria as the Search→ Find option.

### 4.4.4 Search→ Replace

Choose Search→ Replace to search for and replace text within the current buffer. This type of search is always in a forward direction.

When you choose the Replace option, the Replace dialog box shown in Figure 4–10 appears.

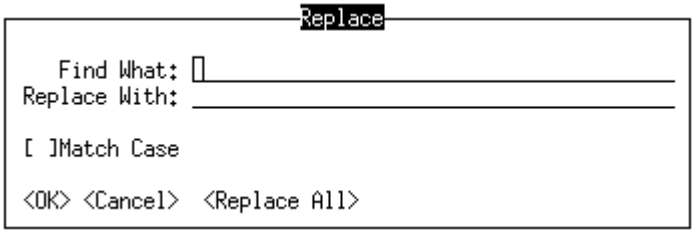



**Figure 4–10: Replace Dialog Box**

Table 4–9 describes the user-interface elements of the Replace dialog box.

**Table 4–9: Replace Dialog Box**

| User-interface Element | Purpose |
|---|---|
| Find What | Identifies the text to search for. |
| Replace With | Specifies the new text you want to insert |
| Match Case | Specifies whether the search is case sensitive. |

The Find What and Replace With fill-in fields default to the text and options you entered the last time you executed the option in the current session.

### 4.4.5 Search→ Goto Line

Choose this option to move the insertion point to a specific line number within the current buffer.

When you choose this option, the Goto Line dialog box appears as shown in Figure 4–11. The Goto Line command lets you enter the line number. It defaults to the line number where the cursor is located in the current buffer. If you enter a number that exceeds the number of lines in the current buffer, Progress moves the cursor to the end of the buffer.

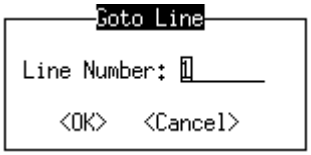



**Figure 4–11: Goto Line Dialog Box**

## 4.5 Buffer Menu

Buffer menu options let you select and view multiple open buffers. Each time you choose File→ New or File→ Open, the Editor creates a buffer containing a copy of a file. Table 4–10 describes the menu that appears when you choose this option.

**Table 4–10: Buffer Menu**

| **Buffer** | |
|---|---|
| List... | Lists the open buffers. |
| Next Buffer | Displays the next open buffer. |
| Previous Buffer | Displays the previous open buffer. |
| Information... | Views information about the current buffer. |

### 4.5.1 Buffer→ List

Choose Buffer→ List to display a list of open buffers. Buffers appear listed in the order in which you open them. When you choose this option, the Buffer List dialog box shown in Figure 4–12 appears.

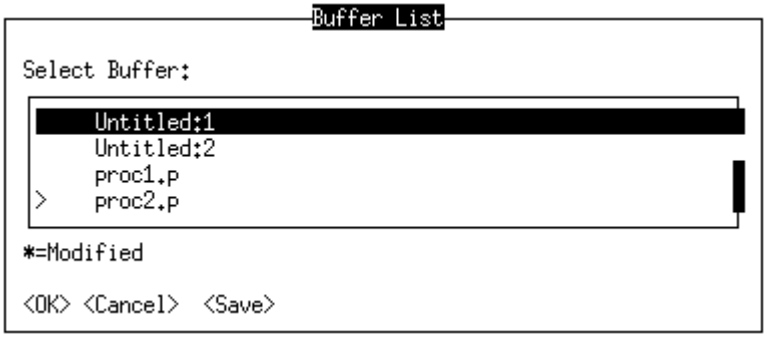



**Figure 4–12: Buffer List Dialog Box**

This list has a marker (>) that indicates the current buffer. You can switch to another buffer by selecting it from the list and pressing **GO**.

An asterisk (*) marks each buffer modified since it was last saved.

### 4.5.2 Buffer→ Next Buffer

Choose this option to display the next buffer in the buffer list. When you choose Buffer→ Next Buffer, the next open buffer appears and becomes the current buffer. Buffers appear in the order in which you open them. If you choose Next when the last buffer in the list is the current buffer, the first buffer in the list appears.

### 4.5.3 Buffer→ Previous Buffer

Choose this option to display the previous buffer in the buffer list. When you choose Buffer→ Previous Buffer, the previous open buffer appears and becomes the current buffer. Buffers appear in the order in which you open them. If you choose Previous when the first buffer in the list is the current buffer, the last buffer in the list appears.

### 4.5.4 Buffer→ Information

Choose Buffer→ Information to view settings for the current buffer. When you choose this option, a Buffer Information dialog box similar to the one shown in Figure 4–13 appears.

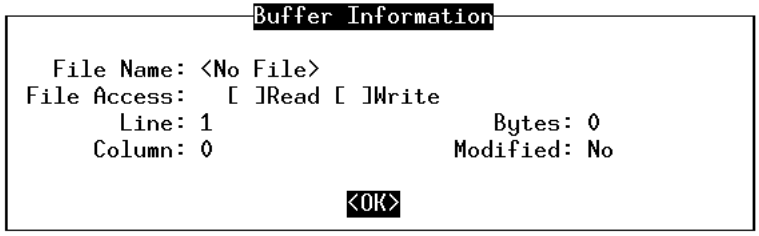



**Figure 4–13: Buffer Information Dialog Box**

Table 4–11 describes the user-interface elements of this dialog box.

**Table 4–11: Buffer Information Dialog Box**

| User-interface Element | Purpose |
|---|---|
| File Name | Displays the name of the operating system file from which the buffer was read. It always displays the full pathname of the file. |
| File Access | Specifies whether the file from which the text in the current buffer was read is read-only or updatable. If the file is read-only, you cannot save changes to the current buffer in the original file. You must choose File→ Save As to save the changes to another file. |
| Line | Displays the line position of the insertion point in the buffer. |
| Column | Displays the column position of the insertion point in the buffer. |
| Bytes | Shows the size of the buffer in bytes. |
| Modified | Indicates whether the buffer has unsaved changes. |

## 4.6 Compile Menu

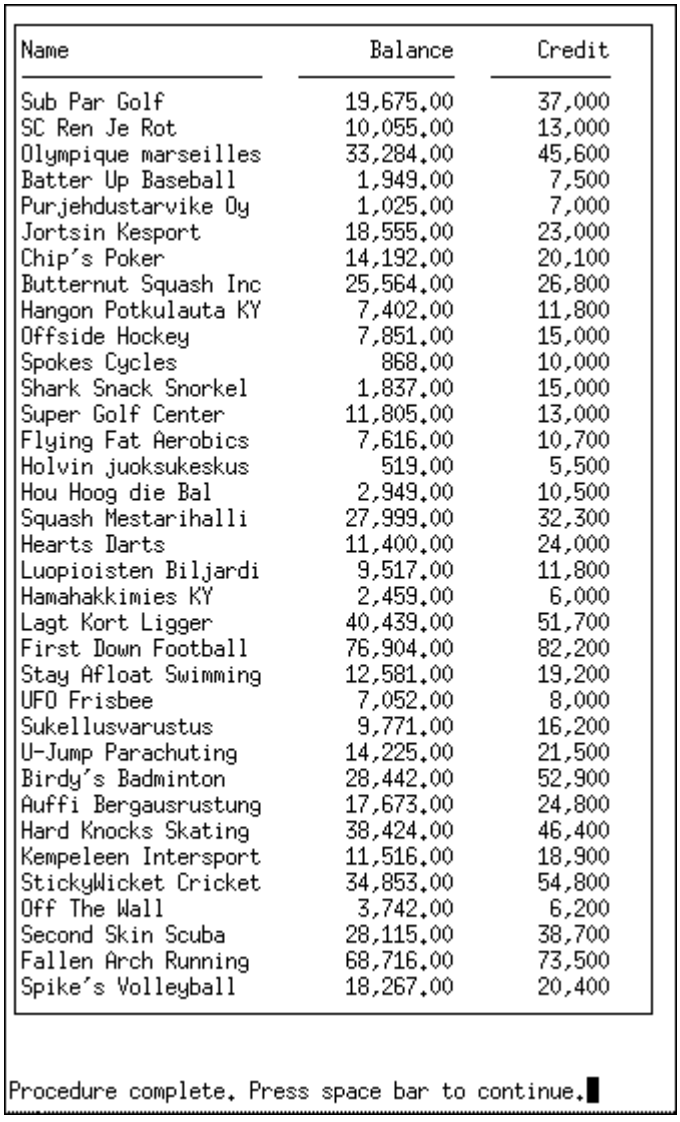
Compile menu options enable you to run and compile Progress procedures. Table 4–12 describes the menu that appears when you choose this option.

**Table 4–12: Compile Menu**

| **Compile** | |
|---|---|
| Run | Compiles and runs the procedure in the current buffer. |
| Check Syntax | Checks the syntax of the procedure in the current buffer. |
| Debug | Invokes the Debugger for the procedure in the current buffer. |
| Compiler Messages... | Displays the compiler messages for the most recent compile. |

### 4.6.1 Compile→ Run

Choose this option to compile and run a procedure in the current buffer. Figure 4–14 shows an example of the screen output of a compiled procedure.



| Name | Balance | Credit |
|---|---|---|
| Sub Par Golf | 19,675.00 | 37,000 |
| SC Ren Je Rot | 10,055.00 | 13,000 |
| Olympique marseilles | 33,284.00 | 45,600 |
| Batter Up Baseball | 1,949.00 | 7,500 |
| Purjehdustarvike Oy | 1,025.00 | 7,000 |
| Jortsin Kesport | 18,555.00 | 23,000 |
| Chip's Poker | 14,192.00 | 20,100 |
| Butternut Squash Inc | 25,564.00 | 26,800 |
| Hangon Potkulauta KY | 7,402.00 | 11,800 |
| Offside Hockey | 7,851.00 | 15,000 |
| Spokes Cycles | 868.00 | 10,000 |
| Shark Snack Snorkel | 1,837.00 | 15,000 |
| Super Golf Center | 11,805.00 | 13,000 |
| Flying Fat Aerobics | 7,616.00 | 10,700 |
| Holvin juoksukeskus | 519.00 | 5,500 |
| Hou Hoog die Bal | 2,949.00 | 10,500 |
| Squash Mestarihalli | 27,999.00 | 32,300 |
| Hearts Darts | 11,400.00 | 24,000 |
| Luopioisten Biljardi | 9,517.00 | 11,800 |
| Hamahakkimies KY | 2,459.00 | 6,000 |
| Lagt Kort Ligger | 40,439.00 | 51,700 |
| First Down Football | 76,904.00 | 82,200 |
| Stay Afloat Swimming | 12,581.00 | 19,200 |
| UFO Frisbee | 7,052.00 | 8,000 |
| Sukellusvarustus | 9,771.00 | 16,200 |
| U-Jump Parachuting | 14,225.00 | 21,500 |
| Birdy's Badminton | 28,442.00 | 52,900 |
| Auffi Bergausrustung | 17,673.00 | 24,800 |
| Hard Knocks Skating | 38,424.00 | 46,400 |
| Kempeleen Intersport | 11,516.00 | 18,900 |
| StickyWicket Cricket | 34,853.00 | 54,800 |
| Off The Wall | 3,742.00 | 6,200 |
| Second Skin Scuba | 28,115.00 | 38,700 |
| Fallen Arch Running | 68,716.00 | 73,500 |
| Spike's Volleyball | 18,267.00 | 20,400 |

Procedure complete. Press space bar to continue.

**Figure 4–14: Example Procedure Output to Screen**

When the procedure is complete, press **SPACEBAR** to return to the Procedure Editor. You return to the buffer, and the cursor is in the same position as when you executed Compile→ Run.

If you run a procedure that contains only the QUIT statement, the procedure stops running and the Procedure Editor automatically exits. If you have buffers open with changes in them, the Procedure Editor prompts you to save the changes before exiting.

### 4.6.2 Compile→ Check Syntax

Choose this option to compile the procedure in the current buffer and check for 4GL syntax errors.

The Compiler Messages dialog box explains any problems in the syntax. Progress displays an alert box when the syntax is correct.

### 4.6.3 Compile→ Debug

Choose this option to access the Debugger tool and debug the procedure code in the current buffer. For more information about debugging a procedure, see the *Progress Debugger Guide*.

**NOTE:** If the procedure in the current buffer does not compile, the Procedure Editor does not open the Debugger tool when you choose this option.

### 4.6.4 Compile→ Compiler Messages

Choose Compile→ Compiler Messages to display the compiler messages for the most recent compile. Figure 4–15 shows an example of a Compiler Messages dialog box.

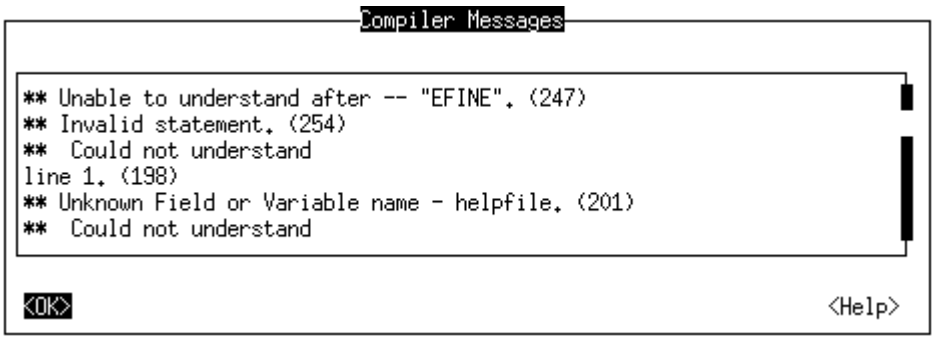



**Figure 4–15: Compiler Messages Dialog Box**

## 4.7 Tools Menu

The Tools menu lets you access other Progress tools. See Chapter 1, "Application Development Environment," for more information about this menu.

## 4.8 Help Menu

Help menu options let you access online help information about Progress error messages and key mappings. See Chapter 1, "Application Development Environment," for more information about this menu.

# 5

# Application Compiler

The Progress Application Compiler utility lets you compile a set of source procedures (.p files) either for the duration of a Progress session (or until Progress runs out of directory entries) or for permanent storage. For a session, Progress creates temporary r-code files. For permanent storage, Progress creates r-code files that inherit the name of the source file and a .r extension by default. Once you compile a procedure, it does not get recompiled when you run it, so the procedure runs quickly. For more information on r-code, see the *Progress Programming Handbook*.

This chapter describes how to use the Application Compiler in the following sections:

- Starting the Application Compiler
- Using the Files/Directories Selection List
- Using the Toggle Boxes
- Using the Action Buttons
- Using the Menu Bar
- Compiling Source Files

## 5.1 Starting the Application Compiler

Choose Tools→ Application Compiler from the menu bar of any other Progress tool to start the Application Compiler.

When you start the Application Compiler, Progress displays the window shown in Figure 5–1.

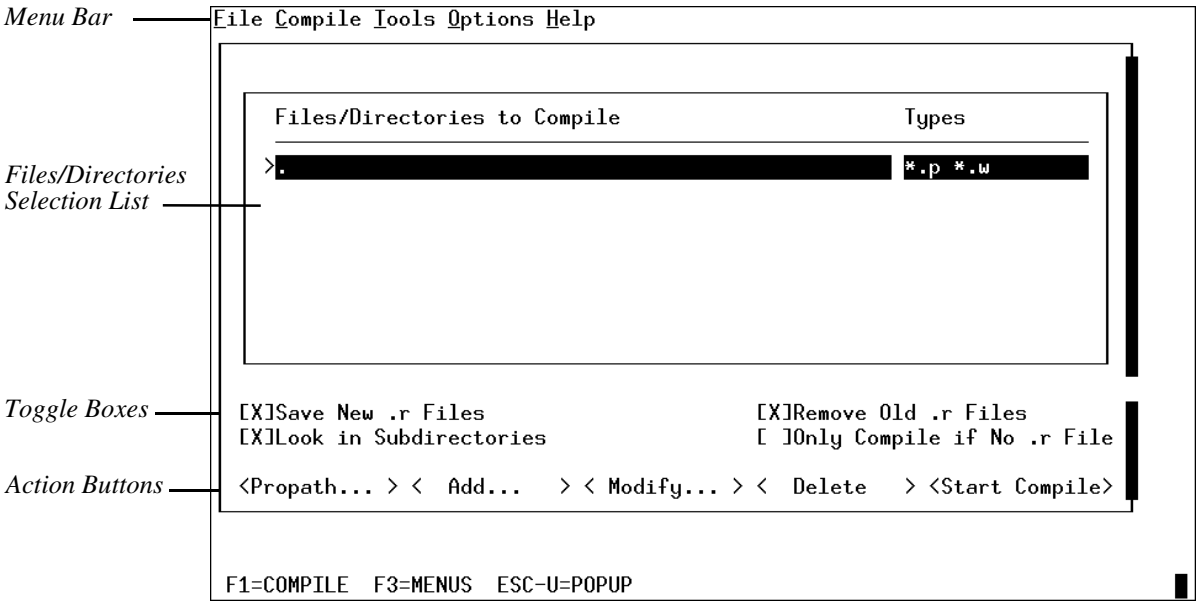



**Figure 5–1: Progress Application Compiler Window**

You use the Application Compiler window to select options and define settings for your compilation.

## 5.2 Using the Files/Directories Selection List

The files/directories selection list displays the files or directories you want to compile. The default list contains only a period (.) indicating the Progress working directory. You can modify the file and directory specifications using the action buttons.

## 5.3 Using the Toggle Boxes

The toggle boxes let you specify criteria for compiling your procedures. The following list describes the toggle boxes and how selecting them affects your compilation:

- **Save New .r Files** — Table 5–1 describes how choosing this toggle box affects the compilation process.

**Table 5–1: The Save New .r Toggle Box and the Compilation Process**

| Action | The r-code File | The Compiler |
|---|---|---|
| Do not select the Save New .r Files | Does not exist | Creates a temporary r-code file that lasts only for the duration of the current Progress session or until Progress runs out of directory entries (-D) and must reuse the directory entry for that r-code file. |
| | Exists | Displays the error message "Compile aborted. SAVE not specified and .r file exists." The message informs you that there is already a valid r-code file, and that the source file is not compiled unless you specify Save New .r Files or Remove Old .r Files. |
| Select the Save New .r Files | Does not exist | Compiles the source file and saves the new r-code file. If errors occur during compilation, no r-code file is created. |
| | Exists | Compiles the source file and, on UNIX and Windows, the new r-code file replaces the existing r-code file. If errors occur during compilation, no r-code file is created, and the existing r-code file remains unchanged. |

If you specify a directory in the Save Into field of the Compiler Options dialog box, the Application Compiler saves the new r-code files into that directory. If you do not specify a directory in the Save Into field, the Application Compiler saves the new r-code files into the directory that contains the source code files.

- **Look in Subdirectories**— Searches for and compiles the specified source file(s) in the subdirectories of the selected directory.
- **Remove Old .r Files** — Deletes any existing r-code file that corresponds to the specified source file in the same directory. You do not need to select this option if you are going to save the new r-code files because newly compiled r-code files automatically replace existing ones.

You must select this option when you want to compile source files for only the Progress session (that is, without saving the r-code files). Thereafter, when you run the compiled procedures, the Application Compiler ensures that you are running the latest versions. If you do not first remove the existing r-code files, the Application Compiler does not recompile the source files. Instead, the error message "Compile aborted. SAVE not specified and .r file exists" appears.

- **Only Compile if No .r File** — Compiles the specified source files only when no corresponding r-code files are found in the selected directory.

**NOTE:** You cannot specify both Remove Old .r Files and Only Compile if No .r File at the same time. When you choose one option, the Compiler automatically deactivates the other.

## 5.4 Using the Action Buttons

The action buttons let you modify the file and directory selection list and compile your files. Table 5–2 describes the action buttons in the Application Compiler window.

**Table 5–2: Compiler Action Buttons**

| Button | Action |
|---|---|
| Propath... | Displays all the directories that are located on your PROPATH. Highlight the directories you want to add to the file/directory selection list and choose OK. |
| Add... | Adds a file or directory outside of the PROPATH to the file/directory selection list. Specify the file or directory name you want to add. |
| Modify... | Modifies the currently selected file or directory specification. Specify the change you want to make to the selected file or directory specification. |
| Delete | Deletes the currently selected file or directory from the file/directory selection list. Verify that you want to delete the selected file or directory from the selection list. |
| Start Compile | Compiles all the files specified in the file/directory selection list. |

When you choose the Add or Modify action buttons, the File Specification dialog box shown in Figure 5–2 appears.

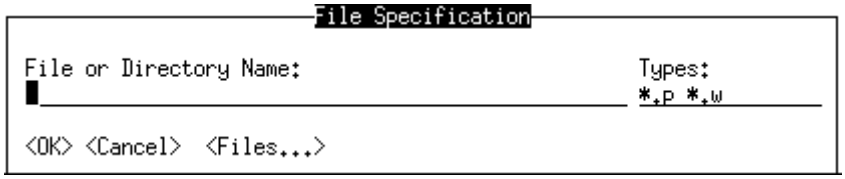



**Figure 5–2: File Specification Dialog Box**

Table 5–3 describes the user-interface elements of this dialog box.

**Table 5–3: File Specification Dialog Box**

| User-interface Element | Purpose |
|---|---|
| File or Directory Name | Specifies the name of the file or directory you want to add or modify. |
| Types | Specifies the filter the Application Compiler uses to perform a match on other strings such as file or directory names. A filter frequently uses a wildcard character to match strings, for instance, a*.* returns all filenames beginning with the letter a and containing a period. The default types are *.p and *.w, which return all file or directory names with the .p and .w extensions. Progress evaluates filters with the MATCHES function. To specify more than one type, separate the types with a space; do not separate them with a comma. |
| Files | Displays the Files dialog box, which lists the files in the currently selected directory. |

## 5.5 Using the Menu Bar

The following sections describe the pull-down menus and options available on the menu bar in the Application Compiler. Table 5–4 describes the menu bar options.

**Table 5–4: Application Compiler Menu**

| Menu | Description |
|---|---|
| File | Exits the Application Compiler. |
| Compile | Compiles Progress source files. |
| Tools | Accesses other Progress tools. |
| Options | Accesses Compiler options and lets you save your Compiler settings. |
| Help | Accesses the Progress Help system. |

### 5.5.1 File Menu

The File menu lets you exit the Application Compiler. Table 5–5 describes the menu that appears when you choose this option.

**Table 5–5: File Menu**

| File | |
|---|---|
| Exit | Ends your current Application Compiler session. |

### 5.5.2 Compile Menu

The Compile menu lets you compile Progress source files. Table 5–6 shows the Compile menu.

**Table 5–6: Compile Menu**

| Compile | |
|---|---|
| Start Compile | Compiles source files. |

### Compile→ Start Compile

When you select this option, the Compiler Results dialog box shown in Figure 5–3 appears.

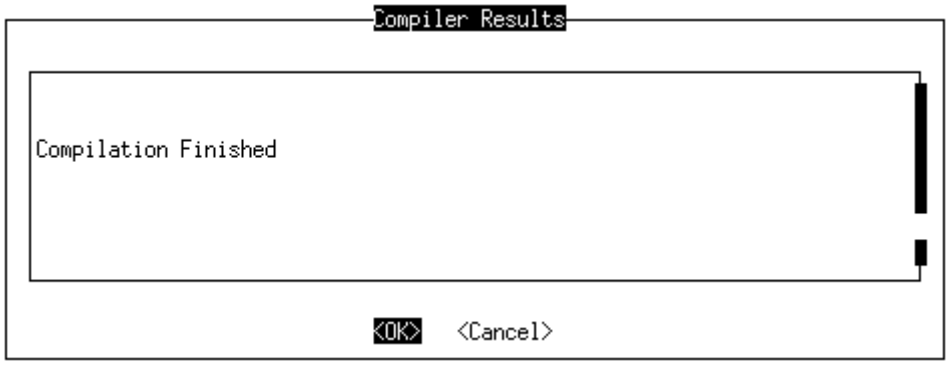



**Figure 5–3: Compiler Results Dialog Box**

The dialog box displays messages about the status of the compilation. If you do not choose Options→ Show Status, the Compiler only indicates when the compilation completes. However, if you choose Options→ Show Status, the Compiler displays all the messages generated during the compilation. For example, the messages state whether a file compiled successfully or whether it failed, and if so, on what line. Other messages include notices of incompatible CRC, etc.

## 5.5.3 Tools Menu

The Tools menu lets you access other tools. See Chapter 1, "Application Development Environment," for more information.

## 5.5.4 Options Menu

The Options menu lets you specify file information and display all the compilation status messages. Table 5–7 lists the Options menu.

**Table 5–7: Options Menu**

| Options | |
|---|---|
| Compiler... | Supplies more information about the source files to be compiled. |
| Show Status | Shows all messages in the Compiler Results window when you compile. |

### Options→ Compiler

Choose Options→ Compiler to supply more information about the file you want to compile. When you choose this option, the Compiler Options dialog box appears as shown in Figure 5–4.

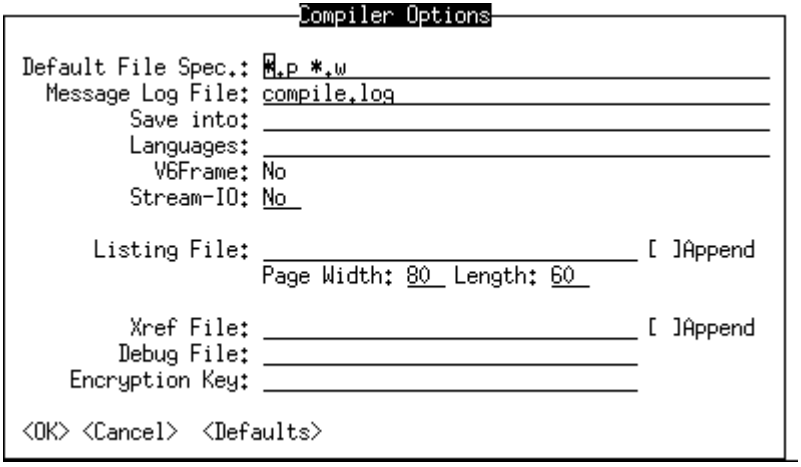



**Figure 5–4: Compiler Options Dialog Box**

Table 5–8 describes the fields in the Compiler Options dialog box.

**Table 5–8: Compiler Options Dialog Box** *(1 of 4)*

| Field | Purpose |
|---|---|
| Default File Spec | Specifies the types of files you want to add to the file/directory specification list. The default file specification is *.p, *.w. |
| Message Log File | Specifies the file to which the Application Compiler sends compiler messages and status. The default output filename is `compile.log`. |
| Save into | Specifies the directory where you want to save the r-code files. By default, the Application Compiler saves the r-code files into the directory that contains the source file. |

**Table 5–8: Compiler Options Dialog Box** *(2 of 4)*

| Field | Purpose |
|---|---|
| Languages | Identifies the language segment to be created in the r-code. The Application Compiler compiles the source file as well as strings translated into different language segments into the r-code file. The default language is the language in your source code. Enter the names of the languages separated by a space. Progress stores translated character strings for each specified language in separate text segments within the r-code file. For the Application Compiler to include translations, you must be connected to a translation database. |
| Stream-IO | Specifies that all output from the compiled procedure be formatted for output to a file or printer. All font specifications are ignored and all frames are treated as if they included the USE-TEXT option. |
| Listing File | Specifies the name of the listing file. The Application Compiler produces a listing of the compilation that includes the following information:<br>• The number of the block where each statement belongs<br>• The name of the source file you are compiling<br>• The date and time at the start of the compilation<br>• The line number of each line in the source file<br>• The complete text of all include files and the name of any subprocedures<br>• The buffer and frame scopes to procedures, internal procedures, and trigger blocks |

**Table 5–8: Compiler Options Dialog Box** *(3 of 4)*

| Field | Purpose |
|---|---|
| Append (Listing File) | Specifies to append the current listing of the compilation to an existing listing file. If you do not select this option, the Application Compiler creates a new listing file that replaces any file of the same name. |
| Page Width | Specifies the page width for the listing file. The default page width is 80 characters per line. Enter a number between 80 and 255. |
| Page Length | Specifies the page length for the listing file. The default page length is 60 lines per page. Enter a number between 10 and 127. |
| Xref File | Specifies the file where the Application Compiler writes cross-reference information between source files and database objects.<br><br>For each object reference, the xref file contains one unformatted and blank-separated line containing the following:<br><br>• Procedure name<br>• Source filename<br>• Line number<br>• Reference type<br>• Object identifier<br><br>See the COMPILE statement in the *Progress Language Reference* for more information on the cross-reference file. |
| Append (xref File) | Specifies to append the xref listing of the source file to an existing xref file. If you do not select this option, the Application Compiler creates a new xref file that replaces any file of the same name. |

**Table 5–8: Compiler Options Dialog Box** *(4 of 4)*

| Field | Purpose |
|---|---|
| Debug File | Specifies the file where the Application Compiler writes a listing to the debug file. The debug file consists of a line-numbered listing of the procedure with the text of all preprocessor include files, names, and parameters inserted. |
| Encryption Key | Specifies the key the Application Compiler uses during compilation to decrypt the source file and any encrypted include files. Encrypted files provide security against users accessing and modifying source files. Progress lets you use either the default key to encrypt source procedures or your own encryption key. If you use your own encryption key, enter it here. If you use an encryption key, the Application Compiler does not produce a listing file as a security measure.<br><br>For more information on encryption keys and procedures, see the COMPILE statement in the *Progress Language Reference* and the XCODE utility in the *Progress Client Deployment Guide*. |
| Defaults | Changes the current Compiler options to the Application Compiler default settings. |

### 5.5.5 Help Menu

Help menu options let you access online help information about the Application Compiler. See Chapter 1, "Application Development Environment," for more information.

## 5.6 Compiling Source Files

Follow these steps to compile source files with the Application Compiler:

1 ♦ Specify the source files you want to compile using the action buttons. You can select files and directories from your PROPATH using the Propath button or from any other directories using the Add button.

2 ♦ Specify the compilation criteria using the toggle boxes.

3 ♦ Choose Options→ Compiler.

4 ♦ Specify any additional configuration information in the fields.

5 ♦ Choose Compile→ Start Compile or press **F1** to compile the source files.

The Compiler Results dialog box appears and displays messages indicating the success or failure of the compilation.

6 ♦ Choose OK from the Compiler Results dialog box to return to the Application Compiler window.

# Index

## A


About Tool option
  Help menu 1–6

Accelerator keys 1–3

Action buttons
  Application Compiler 5–4

ADE. *See* Application Development Environment

_adeevnt.p 3–3

Application Compiler
  action buttons 5–4
  Compile menu 5–6
  compiler options 5–8 to 5–11
  File menu 5–6
  files/directories selection list 5–2
  menu bar 5–6
  Options menu 5–7
  starting 1–3, 5–2
  toggle boxes 5–2
  Tools menu 1–5

Application Development Environment
  Application 1–2

Asterisk (*)
  marker 4–14

Attributes
  buffer 4–15

Audience ix


## B


Bold typeface
  as typographical convention x

Buffer menu
  Procedure Editor 4–13

Buffers
  current buffer name 2–4
  filename 4–15
  information 4–15
  inserting fields 4–9
  inserting files 4–8
  listing 2–5, 4–14
  save as option 4–5
  settings 4–15
  size limits 2–4
  switching 2–5

Bytes
  buffer setting 4–15

## C

Closing
  files 4–5

Column
  buffer setting 4–15

Command keys
  character Procedure Editor 2–4

Compile menu
  Application Compiler 5–6
  compiler messages option 4–18
  Procedure Editor 4–16

Compiler messages 4–18

Compiling
  criteria
    look in subdirectories 5–3
    only compile if no .r files 5–4
    remove old .r files 5–3
    save new .r files 5–2
  source files 5–12

Cursor 2–9

## D

Data Dictionary
  Tools menu 1–5

Debug file
  compiler option 5–11

Default file spec
  Compiler option 5–8

Dialog boxes
  buffer list 4–14
  compiler options 5–8
  compiler results 5–7
  files 4–4
  open files 4–3

## E

Edit menu
  Procedure Editor 4–7

Editor. *See* Procedure Editor

Encryption keys
  compiler option 5–11

Error messages
  displaying descriptions xvi

Example procedures xiv

Exiting
  files 4–5

## F

File menu
  Application Compiler 5–6
  Procedure Editor 4–2
  save as option 4–5

Filename
  buffer setting 4–15

Files
  closing 4–5
  compiling 5–12
  debug 5–11
  exiting 4–7
  listing 5–9
  message log 5–8
  opening 4–3
  xref 5–10

Files/directories selection list 5–2

Find option
  Search menu 4–11

## H

Help
about Tool option 1–6
menu 1–5
Progress messages xvi

## I

Information option
Buffer menu 4–15

Insert fields option
Edit menu 4–9

Insert file option
Edit menu 4–8

Insertion point 2–9
Procedure Editor 2–3

Integration hooks 3–2

Italic typeface
as typographical convention x

## K

Key
encryption 5–11

Keyboard option
Help menu 1–6

Keystrokes x

## L

Languages
compiler option 5–9

Limits
buffer size 2–4

Line
buffer setting 4–15

Listing
buffers 4–14

Listing file append
compiler option 5–10

Listing files
compiler option 5–9

Look in subdirectories
toggle box 5–3

## M

Manual
syntax notation xi

Manual, organization of ix

Markers
asterisk 4–14
current buffer 4–14

Menu bar
Application Compiler 5–6
Procedure Editor 4–2

Menus
Application Compiler
Compile 5–6
File 5–6
Option 5–7
Help 1–5
Procedure Editor
Buffer 4–13
Compile 4–16
Edit 4–7
File 4–2
Search 4–10
Tools 1–4, 5–7

Message log file
compiler option 5–8

Messages
compiler 4–18
displaying descriptions xvi

Messages option
Help menu 1–5

Modifying
buffer settings 4–15

Monospaced typeface
as typographical convention x

Multiple users
startup 1–2, 2–2

## O

Only compile if no .r files
toggle box 5–4

Opening
files 4–3

Options menu
Application Compiler 5–7

OS shell option
Tools menu 1–5

## P

Page length
compiler option 5–10

Page width
compiler option 5–10

Procedure Editor 4–1
Buffer menu 4–13
buffer size limits 2–4
Compile menu 4–16
cursor 2–9
Edit menu 4–7
File menu 4–2
Help menu 1–5
insertion point 2–3
menu bar 4–2
Search menu 4–10
starting 1–2, 2–2
Tools menu 1–4
window layout 2–3

Procedures
examples of xiv

Progress
starting 1–2
multi-user 1–2, 2–2
single-user 1–2, 2–2

## R

R-code file 5–1

Recent messages option
Help menu 1–6

Remove old .r files
toggle box 5–3

Replace option
Search menu 4–12

## S

Save as option
File menu 4–5

Save into
compiler option 5–8

Save new .r files
toggle box 5–2

SCM. *See* Source code management tools

Search menu
Procedure Editor 4–10

Settings
buffer 4–15

Single-user
startup 1–2, 2–2

Source code management tools 3–2

Source files 5–1
compiling 5–12

Starting
Application Compiler 1–3, 5–2
Procedure Editor 1–2, 2–2
Progress 1–2

Stream-IO option
Application Compiler 5–9

Syntax notation xi

## T

Text
find option 4–11
replace option 4–12

Text cursor. *See* cursor

Toggle boxes
look in subdirectories 5–3
only compiler if no .r files 5–4
remove old .r files 5–3
save new .r files 5–2

Tools menu 1–4

Typographical conventions x

## X

Xref file
append
compiler option 5–10
compiler option 5–10