

Progress Results Administration and Development Guide

© 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: 4516
Item Number: 81084;9.1C

Contents

Preface . . . . . **xiii**
Purpose xiii
Audience xiii
Organization of This Manual xiv
How to Use This Manual xiv
Typographical Conventions xv
Syntax Notation xvi
Progress Messages xix
Other Useful Documentation xxi
Getting Started xxi
Development Tools xxiii
Reporting Tools xxiv
4GL xxiv
Database xxv
DataServers xxv
SQL-89/Open Access xxvi
SQL-92 xxvi
Deployment xxvii
WebSpeed xxvii
Reference xxviii
SQL-92 Reference xxviii

1. Results for Administrators and Programmers **1–1**
1.1 Results Product Architecture 1–2
1.1.1 Basic Components 1–2
1.2 Results Development and Administration 1–4
1.3 Starting and Stopping Results 1–4

2. Administering Results 2–1
2.1 The Results Site Administration Menu 2–2
2.2 Results and Application Databases 2–2
2.2.1 Database Configuration Files 2–3
2.2.2 Table Aliases 2–5
2.2.3 Table Relationships and Multi-table Queries 2–7
2.2.4 The Join Construction Dialog Box 2–11
2.2.5 Database Schema Changes 2–13
2.2.6 User Management 2–14
2.3 Site Security Administration 2–16
2.3.1 Application Security 2–17
2.3.2 Data Security 2–20
2.4 Site Defaults 2–25
2.4.1 Standard Page Sizes 2–25
2.4.2 Standard Export Formats 2–26
2.4.3 Standard Label Sizes 2–26
2.4.4 Label Field Selection Defaults 2–28

3. Programming Results 3–1
3.1 Results Architecture and Application Development 3–2
3.2 Application Configuration 3–3
3.2.1 Application Base Files 3–4
3.2.2 Application Preferences 3–5
3.3 Integration Points and Procedures 3–7
3.4 Features 3–10
3.4.1 Feature Procedures 3–11
3.4.2 Feature Editor 3–13
3.4.3 Menu Editor 3–15
3.4.4 Tool Bar Editor 3–18
3.4.5 Feature and Interface Management 3–20
3.5 Results Application Programming Interface (API) 3–20
3.5.1 API Basics 3–20
3.5.2 Accessing and Editing the Current Query 3–26
3.5.3 Generating New Queries 3–40
3.6 Output Solutions in Results 3–43
3.7 Export Solutions in Results 3–45
3.8 Results Help System 3–46
3.9 Results Main Application Procedure 3–48
3.10 Code Generation 3–49
3.10.1 Generating Code from Within Results 3–49
3.10.2 Generating Code from an Application 3–50
3.10.3 Ask at Run Time Values 3–50

4. Deploying Results Applications 4–1
4.1 Deployment Process and Site Requirements 4–2
4.2 Application Files for Deployment 4–2
4.2.1 The Deployment Report 4–3
4.2.2 Other Possible Deployment Files 4–6
4.2.3 Graphics Files 4–6
4.3 Application Databases 4–7
4.4 The Application Directory Structure 4–7
4.5 The Results Help System 4–8
4.6 Application Compilation 4–9
4.7 The Fastload System 4–9

5. Converting Character-based Results Applications to Graphical Results . . . 5–1
5.1 Results Converter Basics 5–2
5.1.1 Converted Information 5–2
5.1.2 Unconverted Information 5–7
5.1.3 Additional Graphical Information 5–8
5.2 Converting Applications 5–8
5.2.1 Starting the Converter 5–8
5.2.2 Converting Only the Configuration File 5–11
5.2.3 Converting Only the Query Files 5–12
5.2.4 Converting Both the Configuration and Query Files 5–13

A. Results Features A–1
A.1 AboutResults A–2
A.2 AdminCustomizeInterface A–2
A.3 AdminDeployment A–2
A.4 AdminExportType A–2
A.5 AdminFeatures A–3
A.6 AdminIntegration A–3
A.7 AdminLabelField A–3
A.8 AdminLabelType A–3
A.9 AdminMenuEdit A–4
A.10 AdminProgInitialize A–4
A.11 AdminProgInstantiate A–4
A.12 AdminProgWrite4GL A–5
A.13 AdminRebuild A–5
A.14 AdminReportType A–5
A.15 AdminReset A–6
A.16 AdminSecurity A–6
A.17 AdminSubMenu A–6
A.18 AdminTableAlias A–7
A.19 AdminTableRelations A–7
A.20 AdminToolBarEdit A–7

A.21 AdminWhere A–8
A.22 Array A–8
A.23 BrowseView A–8
A.24 CalcSubMenu A–8
A.25 Counter A–9
A.26 CustomLabel A–9
A.27 CustomReport A–9
A.28 DataGovernor A–9
A.29 DateFunc A–10
A.30 Exit A–10
A.31 ExportSubMenu A–10
A.32 ExportView A–11
A.33 FieldDelimiters A–11
A.34 FieldProperties A–11
A.35 Fields A–11
A.36 FieldSeparators A–12
A.37 FileClose A–12
A.38 FileDelete A–12
A.39 FileOpen A–12
A.40 FileSave A–13
A.41 FileSaveAs A–13
A.42 FixedWidthFields A–13
A.43 FormView A–14
A.44 FrameProperties A–14
A.45 Generate A–14
A.46 HeadersAndFooters A–14
A.47 HelpContents A–15
A.48 HelpSearch A–15
A.49 HelpUse A–15
A.50 Includes A–15
A.51 Information A–16
A.52 Joins A–16
A.53 LabelView A–16
A.54 LogicalFunc A–16
A.55 Lookup A–17
A.56 MasterDetail A–17
A.57 Math A–17
A.58 Messages A–17
A.59 NewBrowseView A–18
A.60 NewDuplicateView A–18
A.61 NewExportView A–18
A.62 NewFormView A–18
A.63 NewLabelView A–19
A.64 NewReportView A–19
A.65 NewSubMenu A–19

A.66 NumericFunc A–19
A.67 OutputHeaderRecord A–20
A.68 PageBreak A–20
A.69 PercentTotal A–20
A.70 PrintClip A–20
A.71 PrintFile A–21
A.72 PrintPreview A–21
A.73 PrintPrinter A–21
A.74 PrintPrinterNoBox A–22
A.75 PrintSubMenu A–22
A.76 ProgrammingSubMenu A–22
A.77 ReadOtherDirectory A–23
A.78 ReadPublicDirectory A–23
A.79 ReAsk A–23
A.80 RecentMessages A–23
A.81 RecordAdd A–24
A.82 RecordDelete A–24
A.83 RecordEnd A–24
A.84 RecordStart A–24
A.85 RecordUpdate A–25
A.86 ReportView A–25
A.87 RunningTotal A–25
A.88 Selection A–26
A.89 SortOrdering A–26
A.90 StandardExport A–26
A.91 StandardLabel A–27
A.92 StandardReport A–27
A.93 StatusLine A–27
A.94 StringFunc A–27
A.95 Tables A–28
A.96 ToolBar A–28
A.97 Totals A–28
A.98 TotalsOnly A–28
A.99 WriteOtherDirectory A–29
A.100 WritePublicDirectory A–29

B. Results Limits B–1

C. Results Files C–1
C.1 Files Generated by Results C–2
C.2 Editing the results.l File C–4
C.2.1 Delimiter Characters C–4
C.2.2 Standard Export Formats C–5
C.2.3 Standard Label Formats C–7

C.2.4 Standard Page Formats C–8
C.3 Editing a QC7 File C–9
C.4 Editing a QD7 File C–10
C.5 Editing User Interface and Feature Files C–11

Index Index–1

Tables

Table 2–1:	User ID Specifications	2–18
Table 2–2:	User List Examples	2–19
Table 3–1:	Integration Points and Procedures	3–8
Table 3–2:	API Shared Variables	3–21
Table 3–3:	API Procedures	3–23
Table 3–4:	Important Results Features	3–25
Table 3–5:	The Results API and Error Messages	3–26
Table 3–6:	View Character Codes	3–26
Table 3–7:	Calculated Field Symbols	3–31
Table 3–8:	Aggregate String	3–34
Table 3–9:	Sample Output Feature Procedures	3–44
Table 3–10:	Source Files for the Results Help System	3–46
Table 3–11:	Code Generation Using Query→ Generate	3–49
Table 5–1:	File Conversions	5–3
Table 5–2:	Printer Conversion	5–4
Table 5–3:	Query Feature Conversions	5–5
Table B–1:	View Limitations	B–1
Table B–2:	Query Limitations	B–2
Table B–3:	Miscellaneous Limitations	B–3
Table C–1:	Files Generated by Results	C–2
Table C–2:	Definitional Codes for Export Formats	C–5
Table C–3:	Definitional Codes for Label Formats	C–7
Table C–4:	Definitional Codes for Page Formats	C–9

Figures

Figure 1–1: Results Product Architecture 1–2
Figure 3–1: Results Product Architecture 3–2

Procedures

u-mcheck.p 3–16
tstFeat.p 3–42
sport.txt 4–3

Preface

Purpose

This guide introduces you to the administration and development capabilities of Progress Results Version 2 for Windows. Results is an interactive, menu-driven software tool that lets users query, report, and maintain information contained in Progress-supported databases.

Audience

This book is a guide for both Results programmers and site administrators. A Results programmer is a Progress programmer who wants to customize and extend Results. A site administrator is a system administrator at an end user site who has the responsibility of maintaining and configuring Results for end-users. This manual requires that both audiences know and understand the end-user features and functionality of Results. For a basic understanding of the Results Version 2 product, see the *Progress Results User's Guide for Windows* or *Progress Results User's Guide for UNIX*.

Due to the differences between the programming and administration audiences, not all portions of this manual are appropriate for both audiences. The following table suggests a reading path for each audience.

What Do You Want To Learn?	Read This Material:
How to set up and maintain Results.	Chapter 1, "Results for Administrators and Programmers"and Chapter 2, "Administering Results," Appendix A, "Results Features" and Appendix C, "Results Files"
How to develop and deploy a Results application.	All chapters and appendices.

Organization of This Manual

Chapter 1, “Results for Administrators and Programmers”

Provides and overview of administration and development tasks in Results and describes how to start and stop Results.

Chapter 2, “Administering Results”

Describes the system administration tools and procedures to use for a deployed Results application at an end-user site.

Chapter 3, “Programming Results”

Describes the concepts, tools, and programming techniques required to customize and extend Results.

Chapter 4, “Deploying Results Applications”

Explains how to deploy a Results application to an end-user environment.

Chapter 5, “Converting Character-based Results Applications to Graphical Results”

Explains how to use the Results Converter to convert character-based Results applications to graphical Results.

Appendix A, “Results Features”

Contains an alphabetical listing of reference entries for each feature in Results.

Appendix B, “Results Limits”

Lists the limits that exist within Results.

Appendix C, “Results Files”

Provides information about files that Results generates and file formats.

How to Use This Manual

You need not read through the whole manual in detail. A good strategy would be to skim the whole manual once, and then re-read in detail as needed.

Typographical Conventions

This manual uses the following typographical conventions:

- **Bold typefac**e 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 ... ] }
       }
```

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)
```

Use Progress online help to get more information about Progress messages. On the Windows platform, many Progress tools include the following Help menu options to provide information about messages:

- Choose Help→ Recent Messages to display detailed descriptions of the most recent Progress message and all other messages returned in the current session.
- Choose Help→ Messages, then enter the message number to display a description of any Progress message. (If you encounter an error that terminates Progress, make a note of the message number before restarting.)
- In the Procedure Editor, press the **HELP** key (**F2** or **CTRL-W**).

On the UNIX platform, 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 Basic Development Tools (Character only; information for Windows is in online help)

A guide for the Progress development toolset, including the Progress Procedure Editor and the Application Compiler.

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 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.

SQL-92 Reference

(These are non-Progress resources available from your technical bookseller.)

A Guide to the SQL Standard

Date, C.J., with Hugh Darwen. 1997. Reading, MA: Addison Wesley.

Understanding the New SQL: A Complete Guide

Melton, Jim (Digital Equipment Corporation) and Alan R. Simon. 1993. San Francisco: Morgan Kaufmann Publishers.

1

Results for Administrators and Programmers

Results is a powerful query and reporting tool for end users in graphical environments. The power of Results stems from a robust set of default features and a flexible product architecture that allows programmers to customize Results for end-user sites.

This guide focuses on the customization and administration of Results. This chapter serves as a starting point for Results programmers and site administrators. It provides information about the following basic topics:

- Results product architecture
- Results development and administration
- Starting and stopping Results

1.1 Results Product Architecture

An understanding of the architecture of the Results product is important for both programming and administrative activities. The architecture of the Results product is open and modular to support programmer extensions and many different site and user configurations.

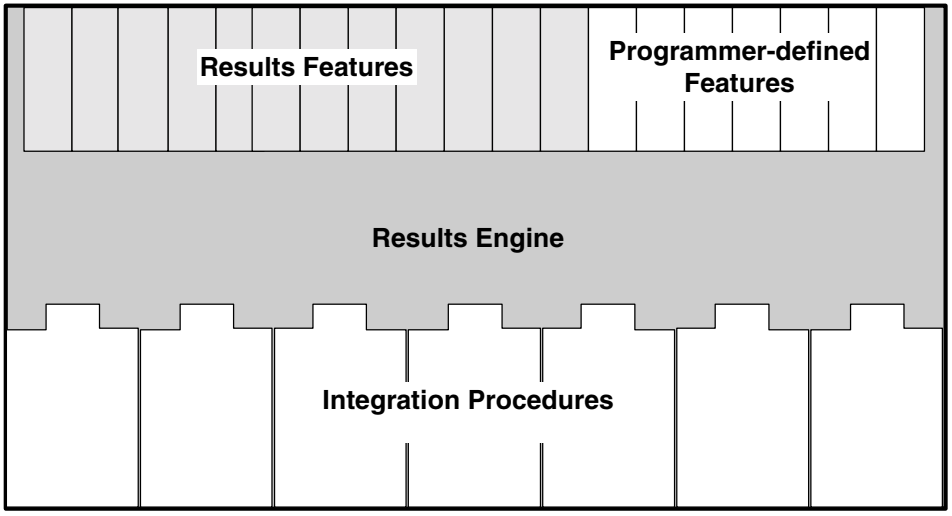


Figure 1–1: Results Product Architecture

1.1.1 Basic Components

The Results product architecture consists of three basic components:

- Features
- Integration procedures
- Results engine

Features

A *feature* is a functional unit in Results that is usually accessed through a menu option or a button on the tool bar. The default Results product contains many features that implement the core functionality of the product. For example, the Query→ Open menu option executes a feature called FileOpen that provides the ability to open a Results query. Programmers can define new features and add them to the Results menu system or toolbar. For example, you can incorporate a graphics package as a feature in Results to graph report data.

Integration Procedures

An *integration procedure* is a Progress procedure that initializes or configures Results for security or site requirements. Each integration procedure is associated with a processing point that occurs during the execution of Results. Each processing point is called an *integration point*. Certain integration points occur only when you start Results and others occur when users perform certain actions in Results. Among other things, integration points facilitate the following:

- Establishment of user identity in Results
- Control of end-user access to database tables and fields in Results
- Control of end-user access to database records in Results queries
- Custom access to Results queries for end users

Programmers can override the default processing at an integration point by associating an integration procedure with the integration point.

Results Engine

The *Results engine* is the software spine that supports the integration, execution, and management of features and integration procedures.

Most programming and administrative tasks focus on features, integration procedures, data access, and interface customization. Results provides several standard features that perform programming and administration tasks, such as:

- Selecting product graphics, icons, and titles for user sites
- Controlling access to features and data on a site or user basis
- Configuring the product menu system and toolbar
- Customizing export formats, report page sizes, label formats, and default label generation behavior for user sites

The Query→ Site Admin and Query→ Customize menus provide access to these standard administrative and programming features in Results. This guide documents the options on these menus and the conceptual information required to use them effectively.

1.2 Results Development and Administration

This guide is for the Results programmer and the site administrator. It is important to understand that these two roles are very different and require different skill sets. The roles are:

- **Results programmer**—A *Results programmer* is a Progress programmer who customizes Results for an end-user site.
- **Site administrator**—A *Results site administrator* is a system administrator at an end-user site who is responsible for maintaining Results and supporting end users.

The structure of this guide and the Results menu system is designed specifically to address these two audiences. The Query→ Site Admin menu option provides access to the features that support the administrative role. Chapter 2, "Administering Results," describes this menu and how to set up and manage users, set up application and database security, and define site defaults.

The Query→ Customize menu option provides access to the features that support the programming role.Chapter 3, "Programming Results," describes this menu and how to customize Results. Chapter 4, "Deploying Results Applications," provides information about how to distribute a Results application.

Along with understanding the programming and administrative roles, it is important to understand the relationship between them. The Results programmer must understand the administration requirements of a Results application and then customize Results to support those requirements. While the programmer might decide to deny all users access to the Customize menu in the user environment, the programmer might grant only the site administrator access to all or a portion of the Site Administration menu. For this reason, Results developers should read Chapter 2, "Administering Results," before customizing Results. By understanding the general Results administration requirements, the developer will be able to address the need of one of the most important users at the end-user site—the site administrator.

1.3 Starting and Stopping Results

Regardless of whether you are a Results programmer or a site administrator, it is important for you to understand how to start Results. This section briefly documents the various ways to start Results and access application databases. For more information about starting and stopping Progress sessions and database servers, see the *Progress Installation and Configuration Guide Version 9 for Windows* or *Progress Installation and Configuration Guide Version 9 for UNIX*.

To start Results, use one of the following startup commands:

Operating System	Syntax
Windows[1]	`prowin` *dbspec* **[** *dbspec* **]** **...** `-p results.p` **[** *parameters* **]**

[1] Modify the Progress Client icon properties to use this command.

If you do not specify a database on command line, Results prompts you to connect to at least one database at startup time. The *dbname* parameter represents the name of a database. All of the startup commands allow you to establish both single-user and multi-user connections to databases. Progress connects to the first database specified after the `prowin` command as a single-user database and connects to the first database specified after the `mprowin` command and the `_prowin` executable as a multi-user database.

The Startup Procedure (-p) parameter specifies the main Progress procedure for the session. The startup procedure for Results is `results.p` located in the Results product directory. The Results product directory must be located in the `PROPATH`. The *parameters* argument represents any other startup parameter required to configure the current Progress session to meet your needs.

You can specify multiple database connections with each of the startup commands. The *dbspec* parameter represents a database connection specification with the following syntax:

```
-db dbname [ -1 ] [ parameters ]
```

The Physical Database Name (-db) parameter allows you to specify the physical name of a database to connect. All databases specified after the first database in a startup command connect as multi-user databases, regardless of the startup command. The Single-user Mode (-1) parameter designates a single-user database connection. You can also specify parameters that describe how Progress connects to a database (for example, identification of the host and server name of a multi-user database on a network).

NOTE: For Results, the recommended setting for the Maximum Memory (-mmax) parameter is 4000.

For more information about the startup parameters referenced in this section, see the *Progress Startup Command and Parameter Reference*.

Before you can establish a multi-user connection to a database, you must start a server for that database. Use one of the following commands to start a Progress server for a database:

Operating System	Syntax
Character	`proserve` *dbname* **[** *parameters* **]**
Windows[1]	`wproserve` **[** *dbname* **]** `-N DDE` **[** *parameters* **]**

[1] Modify the Progress Server Shutdown icon properties to use this command. This command starts an interactive server shell for a database. The server is not a high-performance server and is provided to facilitate testing of multi-user Progress applications in Windows.

As with the startup commands, these server startup commands can also take additional parameters that configure the database server to meet your needs.

Once you start a server process for a database, you cannot establish a single-user connection to that database until you shut down the server. Use one of the following commands to shut down a Progress database server.

Operating System	Syntax
Character mode	`proshut` *dbname* **[** `-by` | `-bn` | `-H` *host-name* | `-S` *server-name* | `-F` **] ...**
Windows[1]	`wproshut` **[** *dbname* **]** **[** *parameters* **]**

[1] Modify the Progress Server Shutdown icon properties to use this command. This command displays a dialog box that allows you to specify how you want to shut down a specified database.

For more information about startup commands, parameters, and database connections, see the *Progress Installation and Configuration Guide Version 9 for Windows* or *Progress Installation and Configuration Guide Version 9 for UNIX*. For more information about database servers, see the *Progress Database Administration Guide and Reference*.

2

Administering Results

This chapter describes system administration tools and procedures for a deployed Results application at an end-user site. It covers the following topics:

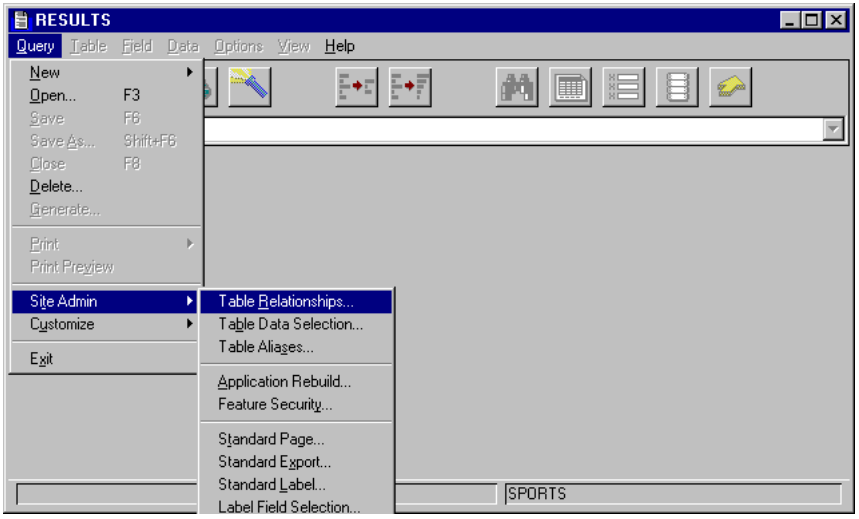
- The Results Site Administration menu
- Results and application databases
- User management
- Security administration
- Site defaults

The information in this chapter is targeted primarily for site administrators; however, programmers who develop and deploy Results applications should also read this chapter. Much of the information presented in this chapter is important for both audiences.

This chapter assumes that site administrators have full access to the Site Admin menu of a deployed Results application. Access to this menu in a deployed Results application is determined by the programmer who developed, configured, and deployed the application. See *Programming Results* for information about how to develop a Results application.

2.1 The Results Site Administration Menu

The focal point for most site administration tasks in Results is the Site Admin menu. To access the Site Administration menu, choose Query→ Site Admin:



This chapter describes how to use the Site Admin menu options to administer a Results application.

2.2 Results and Application Databases

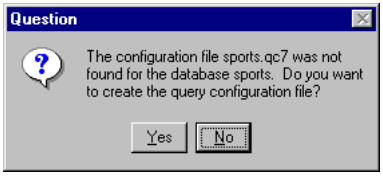
Since Results is a configurable end-user query and reporting tool, it is important for administrators to understand how it relates to application databases. In particular, site administrators should understand the following concepts:

- Results database configuration files
- Database tables and table aliases
- Database table relationships
- Database schema changes

The next few sections in this chapter provide information about these concepts and relationships to help site administrators manage the coexistence of Results with application databases.

2.2.1 Database Configuration Files

When you first run Results against a database, the following message appears:



In this message, sports is the name of the database you are using to start Results. If you are starting Results with more than one database, only the name of the first database connected will appear in this dialog box.

The *initial build* creates a file called `dbname.qc7`. This *query configuration file* (QC7 file) contains the Results default settings for table relationships, security settings, label formats, export formats, and page formats. All the users who use Results with the specified database access this file. Results cannot run without the QC7 file. Therefore, if you are running Results for the first time against a new database or group of databases, Results must perform the initial build to create this file.

The QC7 file is named for the first database connected, so the order in which you connect databases when you start Results is important. For example, if you perform an initial build on the databases db1 and db2 (in that order), the QC7 file for these databases together is called `db1.qc7`. If you start Results with db2 and db1 (in that order), Progress returns a message that the `db2.qc7` file was not found. If you start Results with only the db1 database, a message states that the database configuration has changed and requests you to rebuild the query configuration file.

NOTE: You can start Results with more databases than you used to build the QC7 file. However, the Query→Site Admin→Application Rebuild menu option causes Results to rewrite the entire QC7 file. When Results rewrites a QC7 file, it uses information from all connected databases.

The Location of the QC7 File

Results attempts to create the QC7 file in the same directory that contains the database (*dbname.db*). However, if the database is on another machine, and Results cannot locate that directory, Results places the QC7 file in the current working directory. Once Results creates a QC7 file, you can move it anywhere in your `PROPATH`.

When you start Results, it looks for the QC7 file in the current working directory. If it cannot find it there, it looks in the `PROPATH`. For more information about `PROPATH`, see the *Progress Installation and Configuration Guide Version 9 for UNIX* or *Progress Installation and Configuration Guide Version 9 for Windows*.

Query Configuration Log File

During the initial build and subsequent rewrites of a query configuration file, Results logs information in a query configuration log file (QL file). The name of the query configuration log file is *dbname*.ql, where *dbname* is the name of the first database connected for the session. The information in the query configuration log file can consist of compiler error messages and compilation times. Results creates the QL file in the same directory as the QC7 file.

Results continues to write to the QL file associated with a QC7 file for the life of the QC7 file. If the QL file becomes too large, simply delete the file. Results automatically creates a new QL file.

The results.l File and Initial Builds

Default information for a Results site is stored in the results.l file located in the aderes directory in the Progress product directory ($DLC\GUI). The results.l file is an ASCII file that contains the following default settings:

- Delimiter characters used in report and label format displays
- Definitions for the standard export formats, label sizes, and page sizes available for use within Results
- Definitions for the default report format

Results incorporates the contents of the results.l file into a QC7 file during initial build of the QC7 file. Subsequent rewrites of the QC7 file do not access the results.l file.

You can configure your default site settings by editing the contents of the results.l file prior to performing an initial build. Create a backup copy of the results.l file prior to editing it.

NOTE: If you introduce errors into the results.l file, you might encounter problems during the initial build of a QC7 file.

See Appendix C, "Results Files," for information about the contents of the results.l file.

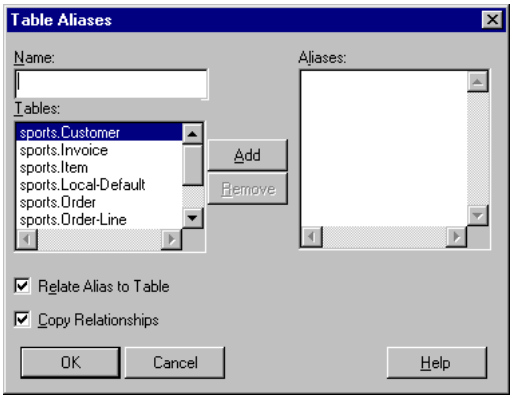
After the Initial Build

After the initial build, you can use the Query→ Site Admin and Query→ Customize menus to override the default environment settings and customize Results for your own site. Most of these menu options edit the content of the current QC7 file. The remainder of this chapter describes how you can customize Results for your site using the Site Admin menu. See Chapter 3, "Programming Results," for information about how you can extend and configure Results using the Customize menu.

2.2.2 Table Aliases

A *table alias* is another name for an existing table in a Results database. Table aliases allow you to perform self-join operations. A *self-join* is when you join a table to itself. For example, the demonstration sports database contains an Item table. Each record in the Item table describes a particular product in a warehouse. You can define an additional field in the Item table to store the item number of a substitute for each item. Once you create the preconditions for the self-join in the schema, you can establish an alias in Results for the Item table called Sub–Item and then join Items with Sub–Items in a Results query.

Choose Query→ Site Admin→ Table Aliases to display the Table Alias dialog box and define an alias for a table:



The Table Alias dialog box contains the following user-interface elements:

Name

Specifies the alias for the selected table.

Tables

Lists the names of all tables available to your user ID in Results from the connected databases.

Aliases

Lists the names of existing aliases. When you select an alias from the list, Results highlights the associated table.

Relationship Buttons

These buttons perform the following actions:

- **Add** — Adds the specified alias in the Name field to the Aliases selection list.
- **Remove** — Removes the currently selected alias.
- **Relate Alias to Table** — Specifies whether to create a self-join relationship between the alias and the selected table. This option is activated by default.
- **Copy Relationships** — Specifies whether the existing table relationships of the selected table also become the defined relationships for the alias. This option is activated by default.

Follow these steps to create a new alias for a table:

1 ♦ Type the name of a new alias in the Name field. The alias name must be unique among all table names and other alias names.

2 ♦ Select a table from the Tables list to associate with the new alias.

3 ♦ If desired, deactivate the Relate Alias to Table toggle box. This option is activated by default.

4 ♦ If desired, deactivate the Copy Relationships toggle box. This option is activated by default.

5 ♦ Choose the Add button to create the alias. The new alias appears in the Aliases list.

Results stores alias definitions in the current QC7 file when you choose the OK button and close the Table Alias dialog box.

You can create several aliases for a single table, but you cannot create an alias for an alias. Once you define an alias for a table, the alias appears as a table in every table listing in Results, and you can create queries that use the alias. To restrict user access to the alias, choose Query→ Site Admin→ Table Data Selection. If you remove an alias that is used in a query, the query will no longer work. If you delete a table, all associated aliases are also deleted.

For more information about restricting user access to tables and aliases, see the "Table Data Selection" section later in this chapter.

2.2.3 Table Relationships and Multi-table Queries

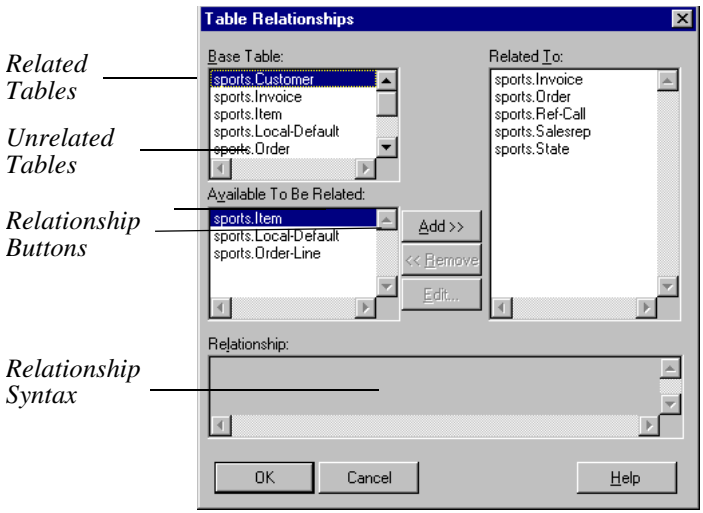
To build a query that contains information from two or more tables, Results must have a description of how the tables used in the query relate to each other. During the initial build process, Results analyzes the connected databases and defines a default set of relationships between tables in the connected databases. A default relationship between two tables is based on the following conditions:

- A field with the same name and data type specification exists in the two tables.
- The field is defined as the only component or the leading component of a unique index in at least one of the tables.

Results stores the default table relationships in the QC7 file following the initial build process.

Default table relationships in Results are called OF relationships in the Progress 4GL. Two tables that meet the conditions stated above can be programmatically joined using the OF option in a Record phrase in a Progress 4GL statement that reads database records. For more information about record access and the Progress 4GL, see the *Progress Programming Handbook*.

Choose Query→ Site Admin→ Table Relationships to display the Table Relationships dialog box where you can view and edit relationships in Results:



The Table Relationships dialog box contains the following user-interface elements:

Base Table

Lists the names of all tables available to your Results user ID from the connected databases. Select a table from this list to access information about the relationships defined for the table in Results.

Available To Be Related

Lists all the tables available to you in Results that are not related to the current base table. Initially, these are the tables that do not meet the OF requirements.

Related Tables

Lists all the tables available that are related to the current base table.

Relationship Buttons

These buttons perform the following actions:

- **Add** — Adds an unrelated table to the list of related tables. If Results cannot form a default relationship between the base table and the selected table, it prompts you to define a relationship.
- **Remove** — Deletes a highlighted table from the list of related tables. During this process, Results deletes the relationship between the highlighted table and the base table, and places the highlighted table to the list of unrelated tables.
- **Edit** — Edits the existing relationship between the current base table and a highlighted table in the list of related tables.
- **Relationship** — Displays the Progress 4GL syntax of the Record phrase that forms the current table relationship. When you highlight a table from the list of related tables, the syntax for the relationship between the current base table and the highlighted table appears. For more information, see the Record Phrase reference entry in the *Progress Language Reference*.

The OK button confirms all modifications made in the Table Relationships dialog box and writes the modifications to the current QC7 file. The Cancel button cancels all modifications made and exits the Table Relationships dialog box.

The default table relationships that Results creates during the initial build process should provide a strong foundation for most multi-table queries. However, you might want to customize the table relationships available to your users. The Table Relationships dialog box allows you to:

- Create new table relationships that do not adhere to the definition of a default table relationship in Results.
- Delete certain default table relationships that provide inappropriate access to sensitive information.
- Redefine certain default table relationships that specify selection criteria or prohibit access to sensitive information.

NOTE: If you change or delete a table relationship, saved queries that depend on that relationship become invalid and no longer function.

Choose Query→ Site Admin→ Application Rebuild to rebuild existing Results queries based on modifications to the table relationships. This option also produces information about any existing Results queries that cannot be rebuilt as a result of modifications to table relationships. See the "Database Schema Changes" section for more information about rebuilding Results queries.

The remainder of this section describes how to use the Table Relationships dialog box to create, edit, and delete table relationships.

Creating a Table Relationship

Follows these steps to use the Table Relationships dialog box to create a new table relationship in Results:

1 ♦ Select a base table for the new relationship. The Table Relationships dialog box displays the current relationship information for that table.

2 ♦ Select a table from the list of unrelated tables.

3 ♦ Choose the Add button. Results attempts to form a default relationship between the base table and the selected unrelated table.

If Results cannot form a default relationship between the two tables, the Join Construction dialog box appears to help you define the relationship. See the "The Join Construction Dialog Box" section for information about defining a table relationship using the Join Construction dialog box.

When you finish building the relationship using the Join Construction dialog box and return to the Table Relationships dialog box, the previously unrelated table appears in the list of related tables and the syntax of the new relationship appears in the Relationship field.

4 ♦ Choose OK in the Table Relationships dialog box to write the table relationship to the current QC7 file.

Editing a Table Relationship

Follow these steps to edit an existing table relationship in Results using the Table Relationships dialog box:

1 ♦ Select a base table.

2 ♦ Select a table from the list of related tables.

3 ♦ Choose the Edit button.

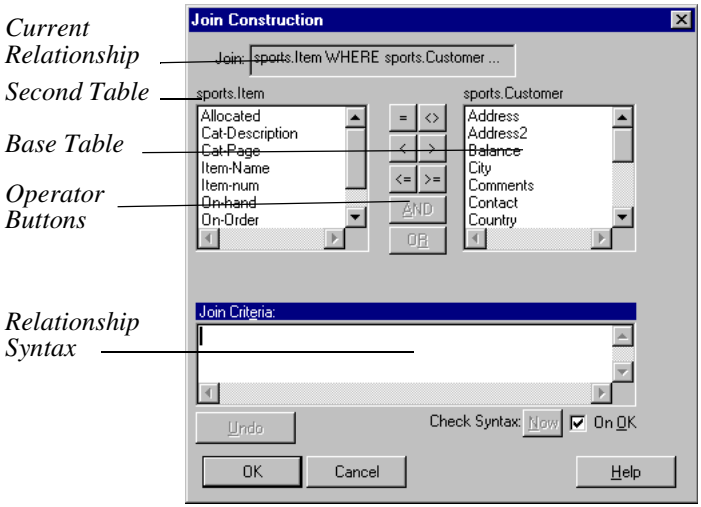
The Join Construction dialog box appears. Use it to redefine the relationship between the current base table and the selected related table. See the "The Join Construction Dialog Box" section for information about defining a table relationship.

When you finish redefining the relationship and return to the Table Relationships dialog box, the syntax of the new relationship appears in the Relationship field.

4 ♦ Choose OK in the Table Relationships dialog box to write the table relationship modifications to the current QC7 file.

2.2.4 The Join Construction Dialog Box

The Join Construction dialog box allows you to build an expression that relates a base table to a second table. The expression that you build with the Join Construction dialog box is called a WHERE clause in the Progress 4GL. Regardless of whether you are adding a relationship or editing a relationship, the Join Construction dialog box is the only tool that supports these tasks. You can only access the Join Construction dialog box from the Table Relationships dialog box:



The Join Construction dialog box contains the following user-interface elements:

Join

Displays the current relationship. If you add a new relationship, the Join field displays the beginning of a WHERE clause that will eventually build a table relationship. If you edit an existing relationship, the Join field displays the relationship. If the existing relationship is an OF relationship, the Customize Join toggle box appears next to the Join field. You must activate Customize Join before you edit an OF relationship between two tables. Editing an OF relationship in the Join Construction dialog box changes the relationship to a WHERE relationship.

Second Table

Displays the fields of the second table.

Base Table

Displays the fields of the base table.

Operator Buttons

Comparison operators that you can use to build an expression between a field in the second table and a field in the base table. The AND and OR buttons represent logical operators that allow you to build compound expressions.

Join Criteria

Specifies the syntax that forms the WHERE relationship between the base and second tables. The following example of a WHERE clause relates a customer table with an order table:

```
WHERE sports.customer.Cust-num = sports.order.Cust-num
```

For more information about the WHERE clause, see the FOR Statement reference entry in the *Progress Language Reference*.

Building a Table Relationship

Follow these steps to define a relationship between two tables in Results using the Join Construction dialog box:

1 ♦ Activate the Customize Join toggle box if it is displayed in the Join Construction dialog box.

2 ♦ Double-click a field from the second table. As you build the expression, the syntax of the expression appears in the Join Criteria area.

3 ♦ Choose a comparison operator button.

4 ♦ Double-click a field from the base table.

5 ♦ If you want to build a more complex relationship, choose either the AND or OR operator button and return to Step 1.

6 ♦ To save the current WHERE relationship and return to the Table Relationships dialog box, choose OK in the Join Construction dialog box.

Deleting a Table Relationship

Follow these steps to delete an existing table relationship from Results using the Table Relationships dialog box:

1 ♦ Select a base table.

2 ♦ Select a table from the list of related tables.

3 ♦ Choose the Remove button.

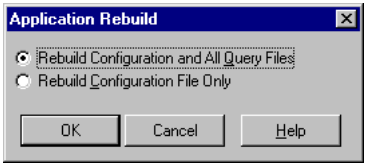
4 ♦ Choose OK from the Table Relationships dialog box to write the table relationship modifications to the current QC7 file.

2.2.5 Database Schema Changes

Database schema changes, such as creating, modifying, or deleting a field, index, or table, can break existing Results queries. Field and table additions and deletions in an application database automatically register in the field and table listings in Results. However, schema modifications do not automatically register in the table relationships maintained in the current QC7 file, or in any existing queries based on those relationships.

Follow these steps to rebuild table relationships and regenerate existing queries based on those relationships:

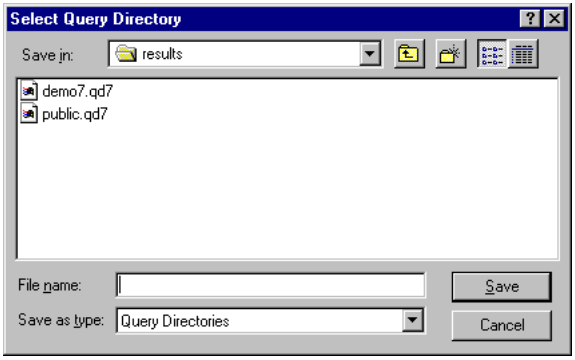
1 ♦ Choose Query→ Site Admin→ Application Rebuild. The Application Rebuild dialog box appears:



The Application Rebuild dialog box lets you rebuild both the table relationships and existing queries or just the table relationships.

2 ♦ To start the rebuild process, select one of the rebuild options, then choose OK.

If you choose to rebuild both the table relationships and existing queries, Results displays the Select Query Directory dialog box, which allows you to select the directory of queries you want to rebuild:



3 ♦ Select the queries you want to rebuild, then choose OK.

4 ♦ When there are no more query directories to rebuild, choose Cancel to end the rebuild operation.

During the rebuild operation, Results writes any problems encountered during the build process to the QL file associated with the current QC7 file. Use this information to identify problem queries that could not be rebuilt by Results. See the "Database Configuration Files" section for more information about QC7 and QL files.

2.2.6 User Management

User IDs let you manage user activities and provide services to users in Results. A *user ID* is a unique identifier assigned to a system user when the user logs onto the system. Most user IDs have an associated password.

Results uses user IDs to create and maintain query directories that store information about the queries that a user creates in Results. The following sections describe user IDs and how Results uses them to create and maintain query directories.

User IDs

User IDs can originate from a variety of sources. Many operating systems support and establish user IDs for system users. On most operating systems, Results can use the operating system user ID as an identifier for a Results user. For more information about user IDs, consult the system administration documentation for your operating system.

To create portable user IDs and passwords in a Progress database, choose Admin→ Security→ Edit User List in the Data Administration tool. A Progress user ID for a user overrides any existing operating system user ID for the user. The user ID definitions must exist in the first database connected for a Progress session. Once a Progress database contains a user definition, an application cannot connect to the database without specifying a valid user ID and password combination. By default, Progress prompts for a valid user ID and password when Results connects to a database that contains user definitions.

There are several other ways to specify a valid user ID and password combination when you connect to a Progress database. See the *Progress Programming Handbook* and the *Progress Installation and Configuration Guide Version 9 for UNIX* or *Progress Installation and Configuration Guide Version 9 for Windows* for more information about Progress user IDs and database connections.

If you are using an operating system that does not support user IDs and there are no defined Progress users in the connected Progress databases, your Results user ID is the "blank user ID." As with any other user ID in Results, you can use feature security to restrict the access of the blank user ID to Results functionality.

Query Directories

A *query directory* is a file that contains up to 256 query definitions for use by one or more Results users. A *query definition* consists of the query name and the name of the databases accessed by the query. All query directory files (QD7 files) have a QD7 file extension.

Results automatically creates a query directory for each user. The QD7 file for a user stores the queries created by that user. Results creates a QD7 file for a user when the user starts Results and there is no QD7 file for that user available in the `PROPATH` or current working directory. The user query directory files are for only one user; they are not multi-user files. The name of a QD7 file for a user takes the form `userid.qd7`. For users who log into Results as the blank user ID, Results creates a generic QD7 file called `results.qd7`. For more information about user IDs, see the "User IDs" section in this chapter.

The public QD7 file serves as a storage place for queries shared by a number of Results users. The name of the public query directory is `public.qd7`, and this name is stored in the current QC7 file associated with the primary application database. There can be only one public QD7 file per QC7 file.

By default, Results creates the public QD7 files in the current working directory. Once Results creates a QD7 file, you can move it anywhere in the `PROPATH`.

Results works with one query directory file at a time. When the user chooses Query→ Save, Results writes the current query to the current QD7 file. A user can open a query from another query directory by choosing Query→ Open or save the current query into another query directory by choosing Query→ Save As. These operations close a previously active query directory and make the query directory specified in the operation the current query directory in Results.

You can control access to the public query directory and other user query directories by setting feature security permissions on the ReadPublicDirectory, ReadOtherDirectory, WritePublicDirectory, and WriteOtherDirectory features. See the "Query Directory Security" section for more information about restricting access to user query directories.

2.3 Site Security Administration

Results supports the following types of security:

- Application Security
- Data Security

The following sections define the concepts and introduce the tools required to configure application and data security settings in Results to meet site requirements. Chapter 3, "Programming Results," contains information about integrating your own application and data security regimes into Results.

2.3.1 Application Security

Application security restricts access to functional units in the Results application and application resources. You can control user access to applications in two ways:

- Feature security
- Query directory security

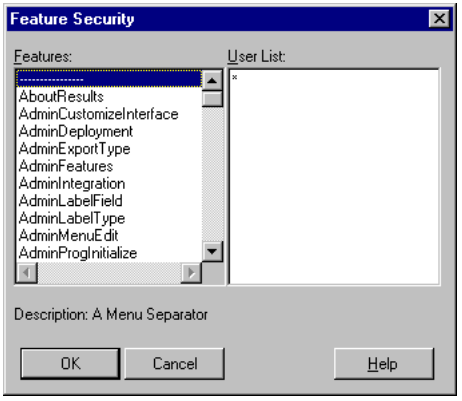
By default, all users can access and use all Results features and can read queries from and write queries to any other query directory or public query directory. However, Results allows you to determine access to features by user ID, and you can protect query directories using Results features. *Features* are functional units in Results that you decide whether to make available to users. Most features are attached to menu options or buttons on the tool bar. See Appendix A, "Results Features," for information about the standard features supplied with the Results product. For information about users and user IDs, see the "User IDs" section in this chapter.

Feature Security

The Results security mechanism reconfigures the application interface for each user based upon that user's permissions. The interface reconfiguration occurs when the user starts Results. When the user does not have permission to use a feature, Results configures the application interface for the user in the following manner:

- Any menu option associated with the feature is removed from the menu bar at Results startup.
- Any icon on the toolbar associated with the feature is removed from the toolbar at Results startup.
- The feature does not appear in any feature listings in the Results product.

Results stores the user permissions for all features in the current QC7 file. To access and change the user permissions for a feature, choose Query→ Site Admin→ Feature Security. The Feature Security dialog box appears:



The features in the Features selection list are either Results core features or add-on features created by your organization or a third party. See Appendix A, "Results Features," for more information about a particular feature in Results.

To set user permissions for a feature, select a feature in the Features selection list, then edit the setting that appears in the User List. A *user list* is a comma-separated list of the user ID specifications. A *user ID specification* is a string expression that specifies one user ID or a group of user IDs.

NOTE: You must define feature for security for all the features. You cannot define one feature and use the defaults for the others.

Table 2–1 shows the forms of a user ID specification.

Table 2–1: User ID Specifications

Specification	Meaning
*	All users can execute the feature.
userid	A specified user can execute the feature.
!*userid*	A specified user cannot execute the feature.
*string**	User IDs that begin with *string* can execute the feature.
!*string**	User IDs that begin with *string* cannot execute the feature.

If the user list for a feature contains multiple user ID specifications, you must separate the user ID specifications with commas. Do not insert blanks between the user ID specifications. Table 2–2 provides some examples of user ID specifications in a user list.

Table 2–2: User List Examples

User List Example	Description
fred,ida,wilma,delmer	Only fred, ida, wilma, and delmer can execute the feature.
!per,!jan,*	Users per and jan cannot execute the feature. Everyone else can.
doc*,dev*	Only user IDs that begin with the strings “doc” and “dev” can execute the feature.
!mark*,!acct*,*	User IDs that begin with the strings “mark” and “acct” cannot execute the feature. Everyone else can.

NOTE: The second and fourth user list examples in Table 2–2 show how to exclude certain users from accessing a feature. It is important to note that the trailing asterisk (*) in both examples allows all other users to access the feature. If you do not specify the trailing asterisk, then no user will be able to access the feature.

When you finish assigning user lists for features, choose OK to write the feature security settings to the current QC7 file and exit the Feature Security dialog box.

Query Directory Security

By default, all users can read queries from and write queries to another user’s query directory or the public query directory. Results provides several features that allow you to determine user access to query directories using the Query→ Open, Query→ Save, and Query→ Save As menu options. These features are: ReadOtherDirectory, ReadPublicDirectory, WriteOtherDirectory, and WritePublicDirectory.

NOTE: Results security does not override any operating system permissions for the query directories.

Use the Feature Security capability to set user permissions for these features and secure user access to query directories. See the previous section, “Feature Security,” for more information about how to set user permissions for a feature.

2.3.2 Data Security

Data security restricts access to data in a connected database to prevent disclosure of sensitive information and to preserve data integrity. Progress provides several security mechanisms to create and enforce data security for an application database. This section describes three types of data security:

- Progress data security
- Table data selection
- Data security features

The Progress data security applies automatically to Results because Results is a Progress application. Site administrators can add their own security on top of Progress—and Results—security using the Query→ Table Data Selection and Query→ Feature Security options.

Programmers can also set up their own security systems specifically for Results using integration points and procedures. See Chapter 3, "Programming Results," for more information about integration points and procedures in Results.

Progress Data Security

Progress allows you to limit compile-time access to tables and fields in a database by user ID. All applications that access an application database, including Results, are governed by the compile-time user access permissions stored in the database. If the users do not have write access to a table or a field in an application database at compile time, they can build queries using the table or field in Results but cannot manipulate the data in the table or field using the form view of a Results query. If users do not have read access to a table or a field in an application database at compile time, that table or field does not appear in their database table or field listings. Also, the users cannot access a Results query that accesses the table or field.

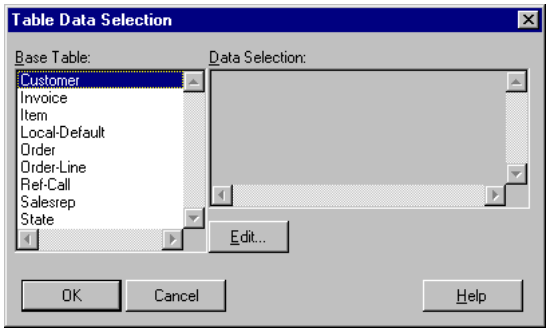
To set access permissions for the tables and fields of a database, access the Data Administration tool, then choose Admin→ Security→ Edit Data Security. These user access permissions for tables and fields are stored as part of the schema in the application database.

For a complete description of data security and security tools in Progress, see the *Progress Database Administration Guide and Reference*, the *Progress Client Deployment Guide*, and the *Progress Programming Handbook*.

Table Data Selection

By default, if users have access to a table, they have access to all records in that table. Site administrators can define selection criteria for a table that functions as a security mechanism to limit access to sensitive data in the table on a record-by-record basis. *Selection criteria* is an expression that determines the records that a query returns from a table.

The Table Data Selection option allows administrators with the ability to define security selection criteria for a table. The selection criteria defined for a table executes as part of all queries that access the table, regardless of user ID. When you choose Query→ Site Admin→ Table Data Selection, the Table Data Selection dialog box appears. Use this dialog box to build selection criteria for a table:



The Table Data Selection dialog box contains the following user-interface elements:

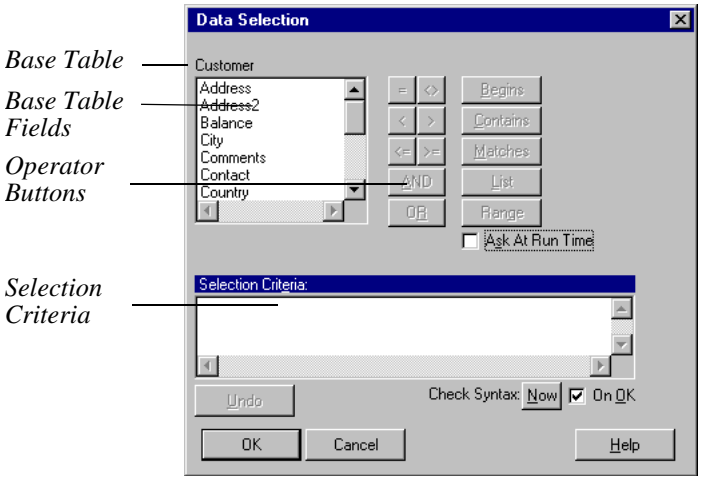
Base Table

Specifies the current table for the definition of selection criteria. When you select a table from this list, any associated selection criteria expression appears in the Data Selection field. By default, tables do not have associated selection criteria expressions.

Data Selection

Specifies an expression that limits the display of records from the current base table. By default, tables do not have selection criteria in Results.

To define or redefine the selection criteria associated with the current base table, choose the Edit button. The Data Selection dialog box appears:



The Data Selection dialog box contains the following information and functionality:

Base Table

Identifies the base table for which you are defining selection criteria.

Base Table Fields

Lists the fields in the base table.

Operator Buttons

Comparison operators that you can use to build an expression for a selected field in the current base table. The operator buttons become sensitive based on the data type of the selected field. Several of these operator buttons allow you to build specialized search criteria:

Operator Button	Search Criteria
Begins	Specifies a search string to match to the beginning of values in the current base field.
Contains	Specifies a text search expression to locate in the text of the current base field. The current base field must be defined as a word index. The text search expression has the following syntax:

SYNTAX

```
word [ & | | | ! | ^ ] word ] ...
```

A *word* is a template string as defined previously for the Matches comparison operator. The ampersand (&) represents a logical AND; a vertical line (|), exclamation point (!), or caret (^) represents a logical OR. These logical operators allow you to build complex text search expressions.

Matches

Specifies a template string to compare against values in the current base field. The template string can be a complete value or a partial value with embedded wildcard characters. A period (.) in a particular position in a string indicates that any single character is acceptable in that position of the string; an asterisk (*) indicates that any group of characters is acceptable, including a null group of characters.

List

Specifies a list of string values to compare with values in the current base field.

Range

Specifies a range of values to compare with values in the current base field.

Ask At Run Time

Controls run-time prompts for key values in the selection criteria. By default, this is not activated and there is no run-time prompt for the selection criteria. When you activate this option, Results requests a prompt string to display in a dialog box prompting for a key value when it executes the current query.

Selection Criteria

Specifies the syntax of the selection criteria expression for the base table.

Check Syntax

Specifies when to check the selection criteria. The Now button specifies to check the syntax immediately; the toggle box indicates to check the syntax when you choose OK. By default, Results checks the syntax when you choose OK.

Follow these steps to define selection criteria for a table using the Data Selection dialog box:

1 ♦ Select a field from the table. As you build the selection criteria expression, the syntax of the expression appears in the Selection Criteria panel.

2 ♦ If you want Results to prompt the user for a value for the specified field at run time, activate the Ask At Run Time toggle box.

3 ♦ Choose a comparison operator button. If Ask At Run Time is not selected, Results prompts for values to form the rest of the expression.

4 ♦ To save the selection criteria for the current base table and return to the Table Data Selection dialog box, choose OK in the Data Selection dialog box.

5 ♦ Choose OK in the Table Data Selection dialog box to write the selection criteria associated with each table to the current QC7 file.

Data Security Features

By default, all users can add, update, and delete records from a table using a Results query in the form view. Results provides several features that allow you to regulate which users can perform these data manipulation activities. These features are RecordAdd, RecordDelete, and RecordUpdate. Use the Feature Security option to set user permissions for these features and to determine which users can manipulate a database using the form view of a Results query.

See the "Feature Security" section for more information about how to set user permissions for a feature.

2.4 Site Defaults

The Site Admin menu provides access to several tools that allow administrators to configure default formats and behaviors used in the creation of query views. These tools allow you to configure the following defaults:

- Standard page sizes
- Standard export formats
- Standard label sizes
- Label field selection defaults

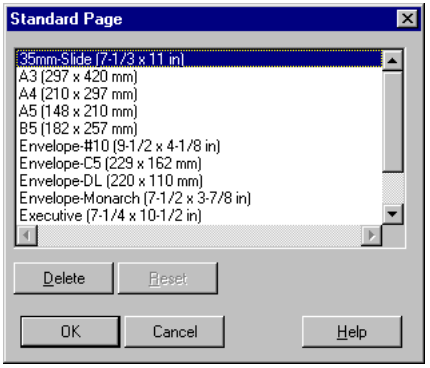
The following sections discuss how to configure these defaults for a Results application.

2.4.1 Standard Page Sizes

The `results.l` file contains definitions for standard page sizes that you can use in report views. During the initial build process, the contents of the `results.l` file is used to create the current QC7 file.

Follow these steps to view and delete paper sizes from the current QC7 file:

1 ♦ Choose Query→ Site Admin→ Standard Page Size. The Standard Page dialog box appears:



2 ♦ Choose Delete to delete a selected paper size.

3 ♦ Choose Reset to restore any page sizes deleted since you opened the dialog box.

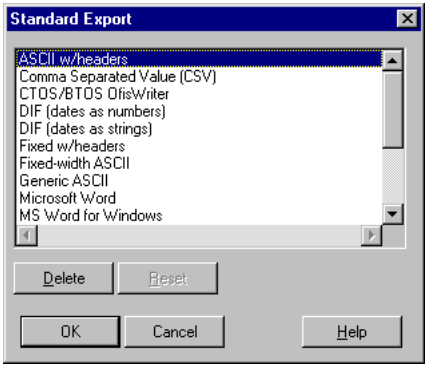
4 ♦ Choose OK to write the changes to the current QC7 file and exit the dialog box.

2.4.2 Standard Export Formats

The `results.l` file contains the definitions for standard export formats that you can use in export views. During the initial build process, Results uses the contents of the `results.l` file to create the current QC7 file.

Follow these steps to view and delete export formats from the current QC7 file:

1 ♦ Choose Query→ Site Admin→ Standard Export. The Standard Export dialog box appears:



2 ♦ Choose Delete to delete a selected export type.

3 ♦ Choose Reset to restore any export type deleted since you entered the dialog box.

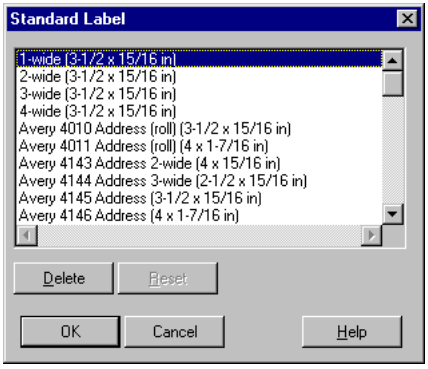
4 ♦ Choose OK to write the changes to the current QC7 file and exit the dialog box.

2.4.3 Standard Label Sizes

The `results.l` file contains definitions for standard label sizes that you can use in label views. During the initial build process, the contents of the `results.l` file is used to create the current QC7 file.

Follow these steps to view and delete label sizes from the current QC7 file:

1 ♦ Choose Query→ Site Admin→ Standard Label. The Standard Label dialog box appears:



2 ♦ Choose Delete to delete a selected label type.

3 ♦ Choose Reset to restore any label type deleted since you entered the dialog box.

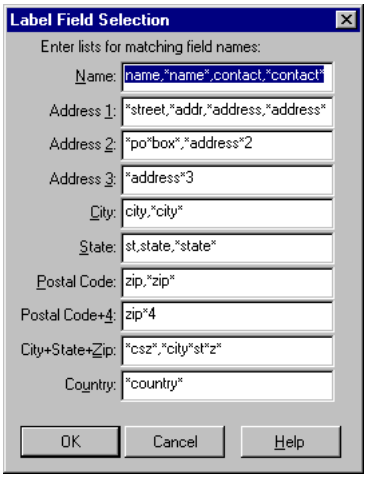
4 ♦ Choose OK to write the changes to the current QC7 file and exit the dialog box.

2.4.4 Label Field Selection Defaults

When a user selects the label view for an active query for the first time, the user can generate either a default mailing label or a custom mailing label for the query.

Follow these steps to display and edit the parameters that Results uses to generate a default mailing label for a query:

1 ♦ Choose Query→ Site Admin→ Label Field Selection. The Label Field Selection dialog box appears:



The Label Field Selection dialog box displays ten possible label fields for default mailing labels. Most of the label fields contain a comma-separated list of strings used to search for database fields in the tables of the active query and generate the default label view for the query.

Results uses the MATCHES function to match a database field name from a table in the active query to a label field. Each string listed in a label field represents a complete field name or a partial field name with wildcard characters. A period (.) in a string indicates that any single character is acceptable in that position; an asterisk (*) indicates that any group of characters is acceptable, including a null group of characters. See the "Label Field Considerations" section for more information about how Results generates labels for queries.

2 ♦ Set up your own label selection parameters that determine how Results generates default labels for your users.

3 ♦ Choose OK to write the changes to the current QC7 file and exit the dialog box.

Label Field Considerations

Before you alter the label selection information in the Label Field Selection dialog box, you must understand how Results generates default labels for queries:

- The order of field names or partial field names specified for a label address field is important. Results attempts to find a matching database field for a label address field starting with the first string specified in the list associated with the label address field and ending with the last string in the list.
- Precedence relationships between label address fields determine the default label layout associated with a query. A database field match with the Postal Code label address field suppresses the display of a database field that matches the Postal Code+4 label address field. A database field match with the City, State, Postal Code, or Postal Code+4 label address field suppresses the display of a database field that matches City+State+Zip label address field.
- For queries based on multiple tables, Results searches for database fields for the default label starting with the last table specified in the query and ending with the first table specified.

Based on this information, you can set up your own label selection parameters that determine how Results generates default labels for your users. However, Results always builds the labels in the format used in the United States.

3

Programming Results

The information in this chapter is for Progress programmers who want to customize Results for distribution in an end-user environment. This chapter contains information on the following topics:

- Results architecture and application development
- Application configuration
- Integration points and procedures
- Features
- The Results Application Programming Interface (API)
- Output solutions
- Export solutions
- The Results help system
- The main application procedure of Results
- Code generation in Results

This chapter describes the concepts, tools, and programming techniques required to customize and extend Results. When you finish reading this chapter, revisit Chapter 2, “Administering Results,” to assess the administration requirements of your extensions.

3.1 Results Architecture and Application Development

From a programming perspective, an understanding of the open and flexible architecture of Results is crucial for success. Most programming activities in Results center around integration points and features. Figure 3–1 shows these integration points and features.

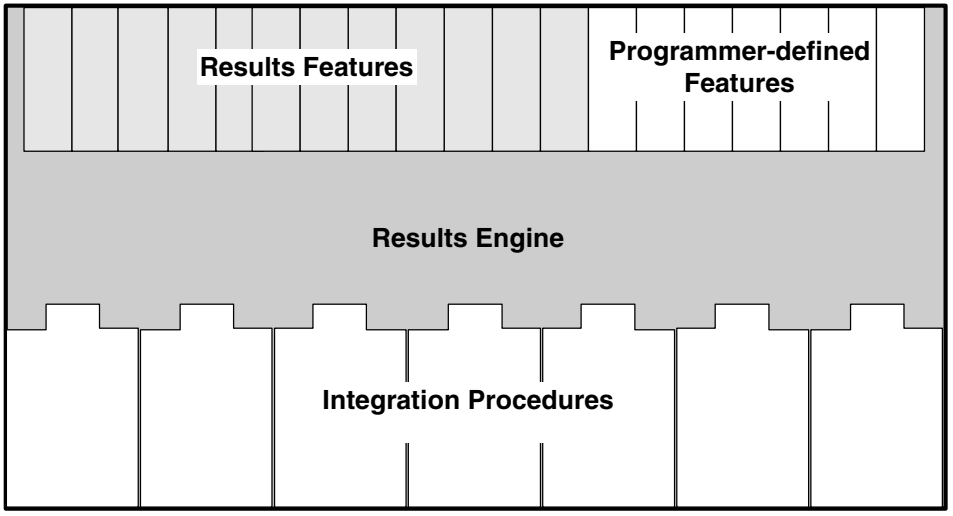


Figure 3–1: Results Product Architecture

An integration point is a processing point used to initialize or configure Results for security or site requirements. Results performs default processing at these points. You can override the default processing at an integration point by associating an integration procedure with the integration point. An integration procedure is a Progress procedure and a feature is a functional unit in the Results product. Results comes with a set of predefined features that implement the core functionality of the product. You can define your own features and add them to the Results menu system or toolbar, or call them from another feature.

The following procedure outlines the basic steps for developing a Results application.

1. Develop or acquire the application databases. The *Progress Database Design Guide* explains how to design and develop an application database.

2. Define the application configuration. The "Application Configuration" section describes how to define an application and configure Results for the development of an application.

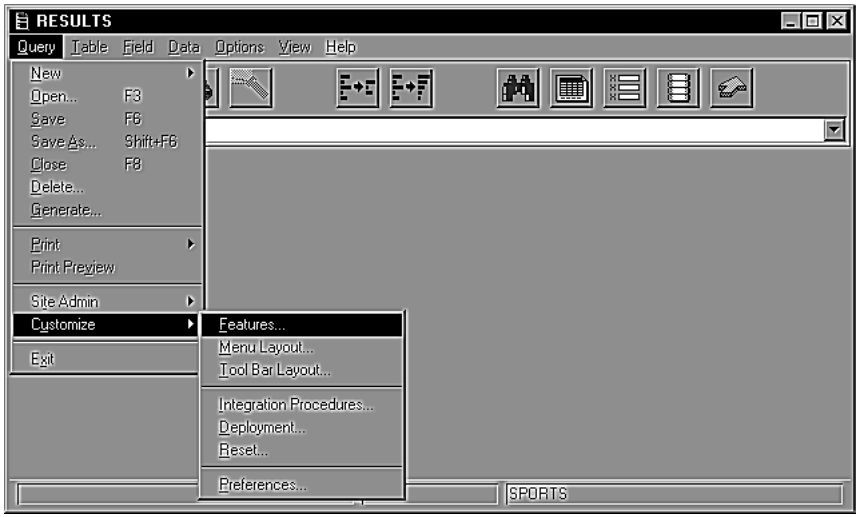
3. Develop integration procedures and integrate them into Results. The "Integration Points and Procedures" section explains how to customize Results using integration procedures.

4. Develop feature procedures and integrate them into Results. The "Features" section explains how to use features to extend Results.

5. Perform any necessary predeployment application administration. For example, you can set up users, query directories, and provide security for the Results application. Chapter 2, "Administering Results," provides more information about the administration tools available in Results.

6. Deploy the application. Chapter 4, "Deploying Results Applications," explains how to deploy a Results application.

The focal point for most application development tasks in Results is the Customize menu. To access the Customize menu, choose Query→ Customize:



The following sections describe how to use the Customize menu to configure Results and develop a Results application.

3.2 Application Configuration

Once you develop or obtain an application database for your Results application, there are two steps you must take to define and configure your Results application.

1. Define the base files for your application in Results. *Base files* are the set of files that define and store configuration information for a Results application.

2. Customize the Results application.

The following sections provide more information about configuring Results and your application.

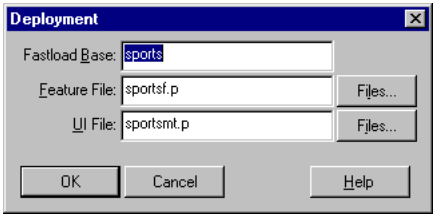
3.2.1 Application Base Files

Before you start to develop feature and integration procedures, define a set of base files to store the configuration information for your Results application. The base files of a Results application are:

- Fastload Files—A set of Progress procedure files that optimize loading information from the current QC7 file into Results at startup.
- Feature File—A Progress procedure file that defines and loads features into Results at startup.
- UI File—A Progress procedure file that defines and loads the layout of the menu system and toolbar into Results at startup.

The definition of a set of base files for your application is a key step in managing the development and deployment of a Results application.

Choose Query→ Customize→ Deployment to access the Deployment dialog box and define base files for an application:



The Deployment dialog box contains the following user-interface elements:

Fastload Base

Specifies a base string so that Results can generate the fastload files associated with the current QC7 file and application. The default base string is the first five characters of the current QC7 file. Each fastload file generated by Results for the current QC7 file and application has a filename that starts with the specified base string and ends with a unique number (for example, `sport###.r`). Limit the base string to five characters. You can specify a pathname with a base string.

Feature File

Enter a filename or choose the Files button to locate and select an existing feature file.

UI File

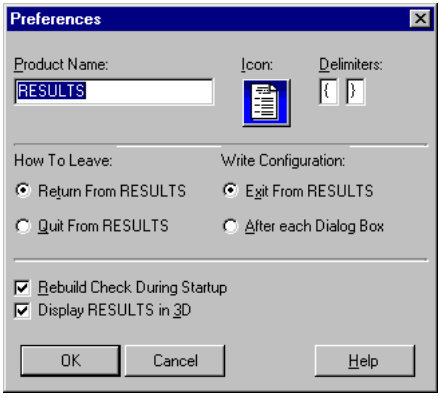
Enter a filename or choose the Files button to locate and select an existing UI file.

You can specify relative or absolute pathnames for the fastload, feature, and UI files. Once Results creates fastload files, you can move them anywhere in PROPATH.

When you start Results, it looks for fastload files in the current working directory. If it cannot find them there, it looks in the directories specified in the PROPATH for the fastload files. For more information about PROPATH, see the *Progress Installation and Configuration Guide Version 9 for UNIX* or *Progress Installation and Configuration Guide Version 9 for Windows*.

3.2.2 Application Preferences

Choose Query→ Customize→ Preferences to display the Preferences dialog box and define default behaviors for your Results application and development environment:



The Preferences dialog box contains the following user-interface elements:

Product Name

Specifies a string to display in the title bar of the Results application window. This string also is displayed in certain error messages.

Icon

Represents the Results application in the current window system. Choose this icon to view and select a graphics file.

Delimiters

Specifies the left and right delimiter characters that delimit functions in report headers and footers and the fields in address (and other) labels. The curly braces { } are the default delimiter characters, but they are not appropriate defaults in some cases. For example, 7-bit character sets use the curly braces for special characters; in this case, specify angle brackets < > as delimiter characters. Do not specify the same character for the left and right delimiters. You can use square brackets [ ] as long as there are no arrays in your reports or labels. You cannot use parentheses ( ) because they interfere with the label-generation code.

How To Leave

Specifies the exit behavior performed by the Exit feature in the Results application:

- **Return From *procedure-name*** — Terminates the Results application and returns to the procedure that called the Results application. If you start the session with `results.p` as the startup procedure, the Progress session terminates. This is the default.
- **Quit From *procedure-name*** — Terminates the Results application and the current Progress session.

Write Configuration

Specifies the configuration save behavior. The options of the Site Administration and Customize menus let you define and modify the configuration information stored in the current QC7 file. This setting allows you to determine when information is written to the current QC7 file

- **Exit From Results**—Saves changes to the current QC7 file when you exit Results. This is the default behavior.
- **After Each Dialog Box**—Saves changes to the current QC7 file when you exit an administration or programming dialog box using the OK button.

When you save the configuration information to the current QC7 file, Results regenerates the fastload files associated with the current QC7 file.

Rebuild Check During Startup

Specifies for Results to check the database schema against the compiled code at startup.

Display Results in 3D

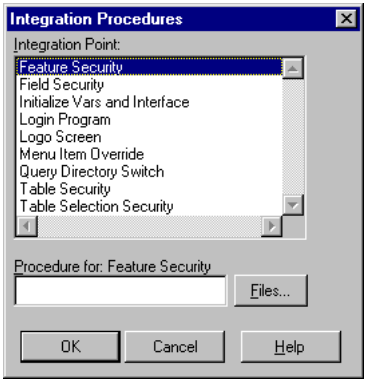
Specifies to display all Results dialog boxes in 3D.

Setting this configuration information allows you to define the Results configuration for the end-user environment and to simulate that configuration in the development environment.

3.3 Integration Points and Procedures

Integration procedures are modules of the Results engine that let programmers integrate Results into an application or end-user environment. These Progress procedures execute at key integration points, such as at Results startup or just prior to executing a query. Results performs default processing at certain integration points and no processing at others. You can insert your own integration procedure at an integration point to override default Results processing.

To define an integration procedure for a Results integration point, choose Query→ Customize→ Integration Procedures. The Integration Procedures dialog box appears:



The Integration Procedures dialog box contains the following user-interface elements:

Integration Point

Lists the available integration points. The current integration point is highlighted.

Procedure for

Specifies the Progress procedure file associated with the current integration point. Choose the Files button to locate an existing procedure file and associate it with the current integration point.

To remove an integration procedure from an integration point, select an integration point and then clear the associated integration procedure name from the Procedure field.

The key to using integration procedures is understanding when they will execute and what is the default processing at the different integration points.Table 3–1 provides important information about integration points and procedures.

Table 3–1: Integration Points and Procedures *(1 of 2)*

Integration Point	Common Use	Execution/ Default Processing	Example Integration Procedure
Feature Security	Prohibits the use of certain Results features by user ID.	Executes once at Results startup. No default processing.	`aderes/u-secure.p`
Field Security	Prohibits access to sensitive fields by user ID. You can use this procedure to specify fields that the user cannot access.	Executes once prior to each time Results displays a list of fields for the user to choose from. No default processing.	`aderes/u-field.p`
Initialize Vars and Interface	Initializes new shared variables and other data structures used by features and integration procedures.	Executes once at Results startup. No default processing.	`aderes/u-define.p`

Table 3–1: Integration Points and Procedures *(2 of 2)*

Integration Point	Common Use	Execution/ Default Processing	Example Integration Procedure
Log Program	Checks user IDs at Results startup for product security configuration.	Executes once at startup. The `aderes/_slogin.p` procedure provides the default processing. You cannot edit `_slogin.p`.	`aderes/u-login.p`
Log Screen	Defines the graphics used in the Results window.	Executes once at Results startup. The `u-logo.p` example procedure provides default processing.	`aderes/u-logo.p`
Menu Item Override	Manages the sensitivity of programmer-defined menu options and toolbar icons.	Executes once after the execution of any feature in Results. No default processing.	`aderes/u-mcheck.p`
Query Directory Switch	Controls the location and user access to query directory (.qd) files.	Executes when the user chooses the Directory button in the Open, Save As, or Delete dialog boxes.	`aderes/u-direct.p`
Table Security	Prohibits access to sensitive tables by user ID. You can use this procedure to specify tables that users cannot access.	Executes once per 350 tables. No default processing.	`aderes/u-direct.p`
Table Selection Security	Prohibits access to sensitive data by user ID. You can use this procedure to insert additional selection criteria into a WHERE clause.	Executes once prior to execution of a query in Results. No default processing.	`aderes/u-where.p`

You can use integration procedures to implement your own data and feature security systems. You can also use integration procedures to instantiate variables and create other preconditions required by features.

The example integration procedure for each integration point documents the parameter definitions and other code structures required at the integration point. The examples can also serve as models for your own integration procedures. For more information about the Results Application Programming Interface, see the "Results Application Programming Interface (API)" section.

3.4 Features

Features are the basic units of functionality in Results. While Results supplies a robust set of features, you might want to define your own features to extend the capabilities of Results.

A Results feature consists of two basic components: a feature procedure and definition. A *feature procedure* is a Progress procedure that implements the functionality of a feature. A *feature definition* describes a feature procedure and information about how the feature procedure is integrated into Results.

There are four basic steps for developing a new feature in Results:

1. Create a feature procedure. See the "Feature Procedures"section for the basic requirements for a feature procedure.

2. Define the feature in the Results product using the Feature Editor. See the "Feature Editor" section for information about how to create a feature definition in Results.

3. If necessary, use the Menu Editor to integrate the feature into the Results menu bar. See the "Menu Editor" section for information about how to integrate a feature into the Results menu system.

4. If necessary, use the Tool Bar Editor to integrate the feature into the Results tool bar. See the "Tool Bar Editor" section for information about how to integrate a feature into the Results toolbar.

See Appendix A, "Results Features," for more information about Results features. The "Feature and Interface Management" section describes how you can manage the integration of features and interface changes on a product-wide basis.

3.4.1 Feature Procedures

Feature procedures are Progress procedures that meet certain information interface requirements for Results. All feature procedures require a character input parameter and a logical output parameter. Consider the following template for a feature procedure:

```
/* Template 1 for a RESULTS Feature Procedure */
/* 1 */
  DEFINE INPUT PARAMETER args AS CHARACTER NO-UNDO.

/* 2 */
  DEFINE OUTPUT PARAMETER winState AS LOGICAL NO-UNDO.

/* Your code here */
```

These notes refer to the commented numbers in the template:

1. Results requires a character input parameter to pass argument values specified in the Argument field of the Feature Editor. All feature procedures must have this input parameter even if you did not specify argument values in the Argument field.

   The Progress 4GL supplies several language elements that allow you to parse strings, parse lists, and convert character values into different data types. For more information about these and other language elements, see the *Progress Language Reference*.

2. This logical output parameter determines whether the Results window redisplays when the feature procedure finishes executing. If the feature procedure changes the current query or current view of the query, set the value of this output parameter to TRUE.

You can code feature procedures to perform all types of processing including interactive and noninteractive processing. Results supplies a complete API that you can use within a feature procedure to access and manipulate information. The Results API is based on a set of procedures and shared variables that contain information about the current state of the Results application and the current query in Results. For more information about the Results API see the "Results Application Programming Interface (API)" section.

Feature procedures can manipulate the Results application window, display a dialog box, or display a secondary application window. To implement a feature procedure that displays a second window, use the following technique to manage user focus and multiple windows in Results:

```
/* Template 2 for a RESULTS Feature Procedure - Multi-windows */

  DEFINE INPUT PARAMETER args AS CHARACTER NO-UNDO.
  DEFINE OUTPUT PARAMETER winState AS LOGICAL NO-UNDO.

/* 1 */
  DEFINE VARIABLE resultsWin AS WIDGET-HANDLE NO-UNDO.
  DEFINE VARIABLE newWin AS WIDGET-HANDLE NO-UNDO.

/* 2 */
  ASSIGN resultsWin = CURRENT-WINDOW

/* 3 */
    resultsWin:WINDOW-STATE = WINDOW-MINIMIZED
    resultsWin:SENSITIVE = FALSE.
  CREATE WINDOW newWin.
  CURRENT-WINDOW = newWin.

  /* Your code here */

/* 4 */
  DELETE WIDGET newWin.
  CURRENT-WINDOW = resultsWin.
  ASSIGN
    resultsWin:WINDOW-STATE = WINDOW-NORMAL
    resultsWin:SENSITIVE = TRUE.
```

The commented numbers in the code refer to the notes below:

1. Create a variable to hold the widget handle of the Results window.

2. Place the widget handle of the Results window into the new variable. The CURRENT–WINDOW session handle returns the widget handle of the Results window.

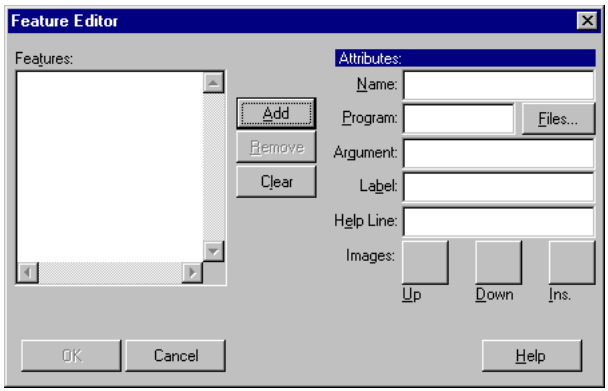
3. Minimize the Results window and disable it for user input while the feature procedure displays the second window.

4. Restore the Results window and enable it for user input after the user closes the second window.

See the "Generating New Queries" section for a sample feature procedure. See the *Progress Language Reference* for a reference entry for each language element in the Progress 4GL.

3.4.2 Feature Editor

The Results Feature Editor allows you to define, modify, and delete features in Results. To access the Feature Editor, choose Query→ Customize→ Features. The Feature Editor dialog box appears:



The Feature Editor dialog box contains the following user-interface elements:

Features

Lists the programmer-defined features. As you define features, this list grows. Select a feature from this list to establish a current feature that you can copy, modify, or remove.

Feature Edit Buttons

Allow you to do the following:

- **Add** — Defines a new feature in Results.
- **Remove** — Removes the current feature. You cannot delete a feature from Results if it is associated with a menu option or a toolbar icon. When you remove a feature, the procedure file associated with the feature is **not** deleted from your operating system.
- **Clear** — Clears the values specified in the Feature Editor. This action does not affect the definition of the current feature.

Name

Specifies the name of the current feature. You cannot change the name of an existing feature.

Program

Specifies the name of a Progress procedure associated with the current feature. Choose the Files button to locate an existing procedure file and associate it with the current feature. The next section provides a general framework for the definition of a feature procedure.

Argument

Specifies the argument values to pass to a procedure identified in the Program field in comma-separated list.

Label

Specifies the default menu label for the current feature. This label appears by default when you add the feature to the Results menu system. You can specify a mnemonic for the label by placing an ampersand (&) before a character within the label.

Help Line

Specifies a help string to associate with a toolbar representation of the feature. This help string appears at the bottom of the Results application window when the user right-clicks the icon associated with the feature.

Images

Identifies the default graphics files that represent the following icon states for a representation of the feature on the toolbar:

- **Up** —The unselected, sensitive state of the icon.
- **Down** — The selected state of the icon.
- **Ins.** — The unselected, insensitive state of the icon.

To specify a graphics file for an icon state, choose one of these buttons. Results displays a dialog box that allows you to locate and browse graphics files in your operating system, view them, and select one. In Windows, the graphics file must be a Windows Bitmap (.bmp) or Icon (.ico).

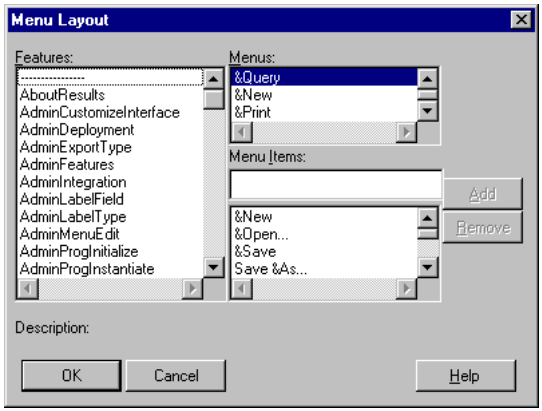
When you finish defining a feature, you can add it to the Results menu system or toolbar. You can also use the Feature Security option described in Chapter 2, “Administering Results,” to control end-user access to features.

3.4.3 Menu Editor

Results provides a dynamic menu editor that you can use to customize user access to Results and programmer-defined features. Once you define a feature, you can add it to an existing menu.

NOTE: You cannot add a menu to the Results menu system; you can only add a menu option to an existing menu. You cannot delete a main menu (for example, Query or Help).

To access the Menu Editor, choose Query→ Customize→ Menu Layout. The Menu Layout dialog box appears:



The Menu Layout dialog box contains the following user-interface elements:

Features

Lists all the features in Results. When you select a feature, the menu and menu option lists scroll to display an associated menu option and/or menu. If the feature is not in the menu system, you can specify a new label for the feature and add it to the selected menu in the menu list. The first entry in the features list is a horizontal rule that you can add to a menu as a separator.

Menus

Lists the menus available in the Results menu system. When you select a menu, the menu options of the selected menu appear in the menu option list. You can then select and add a feature to the menu or select and remove a menu option from the selected menu.

Menu Items

Lists the menu options available on the selected menu in the menu list. Once you select a menu option, you can use the Remove button to remove it from the selected menu. You can also modify the menu item label.

Menu Edit Buttons

Allow you to do the following:

- **Add** — Adds a new menu option to the selected menu in the menu list using the feature selected in the features list.
- **Remove** — Deletes the selected menu option from the selected menu in the menu list.

The following section describes how to define menus and implement menu management strategies in Results.

Menu Management

Results automatically manages the display state of default menu options and menus in the menu system. In general, there are some default menus and menu options that are sensitive only when there is an active query in Results. These menus and menu options become insensitive when the user closes the active query. For example, the Table menu is sensitive only when there is an active query. And while the Query menu is always active by default, there are some menu options on this menu (for example, Save) that are sensitive only when there is an active query.

An understanding of this default menu management behavior in Results is crucial when adding menu options. Before adding menu options to Results, assess the menu management behavior and plan any additions based upon when you want users to access the new menu elements. For example, to add a menu option that is always available to a user, add the menu option to the Query menu.

By default, all menu elements that you add to the menu system are sensitive. You can override the default menu management behavior and also manage the sensitivity of additions to the Results menu system using the Menu Item Override integration point. A Progress procedure attached to this integration point executes once after the execution of any feature in Results. The `u-mcheck.p` procedure located in the `aderes` product directory is the example code template for this integration point:

u-mcheck.p

```
/* Instructional Comments */

  DEFINE INPUT         PARAMETER featureList       AS CHARACTER NO-UNDO.
  DEFINE INPUT-OUTPUT  PARAMETER sensitivityList   AS CHARACTER NO-UNDO.
  DEFINE INPUT-OUTPUT  PARAMETER checkList         AS CHARACTER NO-UNDO.

/* Your Code Here */

  RETURN. /* Required last line */
```

The following list describes the parameters required by a procedure associated with the Menu Item Override integration point. These parameters provide information about all menu options and menus in the Results menu system and buttons on the Results toolbar:

`featureList`

This input parameter is a comma-separated list of feature names associated with menu options in the Results menu system and buttons on the Results toolbar.

`sensitivityList`

This input-output parameter is a comma-separated list of TRUE or FALSE values. Each entry in this list represents the current SENSITIVE attribute setting for a menu option or button associated with a feature listed in the featureList parameter. The order of the sensitivity settings in sensitivityList parameter corresponds to the order of features listed in the featureList parameter. You can set the state of the SENSITIVE attribute for a menu option or button in the integration procedure.

`checkList`

This input-output parameter is a comma-separated list of TRUE or FALSE values. Each entry in this list represents the current CHECKED attribute setting for a menu option associated with a feature listed in the featureList parameter. The order of the CHECKED settings in checkList parameter corresponds to the order of features listed in the featureList parameter. If the current menu option widget is a toggle menu option, you can set the CHECKED attribute of the widget in the integration procedure.

Consider the following code example:

```
/* Example u-mcheck.p */

  {aderes/u-pvars.i}

  DEFINE INPUT         PARAMETER featureList       AS CHARACTER NO-UNDO.
  DEFINE INPUT-OUTPUT  PARAMETER sensitivityList   AS LOGICAL   NO-UNDO.
  DEFINE INPUT-OUTPUT  PARAMETER checkList         AS LOGICAL   NO-UNDO.

  DEFINE VARIABLE i AS INTEGER NO-UNDO.

  DO i = 1 TO NUM-ENTRIES(featureList):
    IF ENTRY(i, featureList) = "myItem" AND qbf-module = "r" THEN
      ENTRY(i, sensitivityList) = FALSE.
  END.

  RETURN. /* Required last line */
```

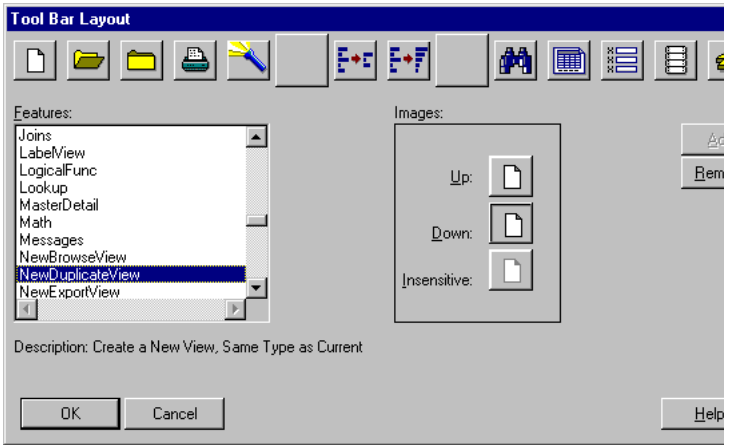
This example procedure checks the featureList for a feature name. If the feature name exists in the featureList and the current view in Results is the report view, the sensitivityList setting for the feature is FALSE. The "Results Application Programming Interface (API)" section provides more information about the qbf–module variable and the `u-pvars.i` include file. For more information about the ENTRY statement, RETURN statement, CHECKED attribute, and SENSITIVE attribute, see the *Progress Language Reference*.

3.4.4 Tool Bar Editor

Results also provides a dynamic Tool Bar Editor that you can use to customize user access to Results and programmer-defined features. Once you define a feature, you can add it as an icon on the toolbar.

To access the Tool Bar Editor, choose Query→ Customize→ Tool Bar Layout. The Tool Bar Layout dialog box appears:

Tool Bar



The Tool Bar Layout dialog box contains the following user-interface elements:

Tool Bar

Represents the tool bar. Select an icon to establish it as the icon in the editor you want to edit.

Features

Lists of all features in Results. Select a feature from this list to add to the toolbar as an icon.

Images

Specifies the default graphics files that represent the following icon states for a representation of the feature on the toolbar:

- **Up** — The unselected, sensitive state of the icon.
- **Down** — The selected state of the icon.
- **Insensitive** — The unselected, insensitive state of the icon.

To specify a graphics file for an icon state, choose one of these buttons. Results displays a dialog box that allows you to locate and browse graphics files in your operating system, view them, and select one. In Windows, the graphics file must be a Windows Bitmap (.bmp) or Icon (.ico).

Tool Bar Edit Buttons

Allow you to do the following:

- **Add** — Adds the selected feature with associated icons to the tool bar at the selected icon location.
- **Remove —** Deletes the selected icon from the tool bar.

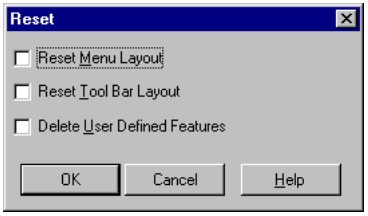
Tool Bar Management

Results automatically manages the display state of default buttons on the tool bar. Most of the default buttons on the Results tool bar are sensitive only when there is an active query in Results. These buttons become insensitive when the user closes the active query. For example, the Save button is sensitive only when there is an active query.

By default, depending on which feature you are using, not all buttons that you add to the Results tool bar are sensitive. You can override the default tool bar management behavior and also manage the sensitivity of added icons using the Menu Item Override integration point. A Progress procedure attached to this integration point executes once after you execute any feature in Results. See the "Menu Management" section for more information about the Menu Item Override integration point.

3.4.5 Feature and Interface Management

One of the most powerful features of Results allows you to extend and customize it. Choose Query→ Customize→ Reset to access the Reset dialog box:



The Reset option allows you to reset the menu system, tool bar, and features of Results to their default states.

From this dialog box, you can reset the Results menu system, toolbar, feature set, or all three. When you choose OK, the reset operation is permanent and your changes to the Results interface are lost.

NOTE: You must remove features from the menu system and tool bar before you can delete them from Results.When you delete a feature, the procedure file associated with the feature is **not** deleted.

3.5 Results Application Programming Interface (API)

While you can code feature procedures to do almost anything you want, most programming in these procedures centers around the following tasks:

- Accessing and editing the current query definition in Results
- Generating new queries in Results

You can also execute Results features programmatically from an integration procedure or another feature procedure. Results provides a robust Application Programming Interface (API) that supports these activities. The following sections provide some basic information about the Results API and describe how to access, modify, and create queries in Results.

3.5.1 API Basics

The Results API consists of a set of shared variables and a procedural interface. The following sections provide basic information about how to access the shared variables and procedures of the Results API in a feature or integration procedure.

Accessing Shared Variables

Results establishes and maintains a set of shared variables, called API shared variables, that you can access and set from a feature or integration procedure. *API shared variables* provide information about the current state of the Results application and the current query in Results. Table 3–2 provides summary information about the API shared variables available in Results.

Table 3–2: API Shared Variables *(1 of 2)*

Shared Variable Name	Data Type	Extent	Description
qbf–vers	CHARACTER	N/A	A version number for the current installation of Results.
qbf–module	CHARACTER	N/A	A character code representing the current Results view.
qbf–name	CHARACTER	N/A	The name of the current query in Results.
qbf–sortby	CHARACTER	N/A	A comma-separated list of field names to sort by. The fields are listed by precedence in the sort.
qbf–count	INTEGER	N/A	An integer representing the number of records returned by the current query.
qbf–governor	INTEGER	N/A	An integer representing the number of records to process in report, label, and export views. Zero (0) processes all records.
qbf–rc#	INTEGER	N/A	The number of fields (database, calculated, and arrays) in the current query.
qbf–rcn	CHARACTER	64	The name of each field in the current query and additional expression information.
qbf–rcc	CHARACTER	64	Information about each calculated field and stacked array in the current query.
qbf–rcg	CHARACTER	64	Information about aggregate fields in the query.

Table 3–2: API Shared Variables *(2 of 2)*

Shared Variable Name	Data Type	Extent	Description
qbf–rcl	CHARACTER	64	Labels for each field in the current query.
qbf–rcf	CHARACTER	64	Formats for each field (database or calculated) in the current query.
qbf–rcp	CHARACTER	64	A comma-separated list of values for each field representing layout information for the field in a report, form, and label views.
qbf–rcw	INTEGER	64	The width in characters for each field in a report view of the current query.
qbf–rct	INTEGER	64	Data type identifiers for each field in the current query. The integer identifier corresponds to a list entry in the qbf–dtype shared variable.
qbf–dtype	CHARACTER	N/A	A comma-separated list of data types.

To access these API shared variables in a feature or integration procedure, you must define them in your procedure. Typically, you use the DEFINE SHARED VARIABLE statement to define a previously established shared variable for use in a procedure. However, Results provides the `u-pvars.i` include file to define the API shared variables for you. Use the following syntax to reference the `u-pvars.i` include file in the feature or integration procedure:

```
{aderes/u-pvars.i}
```

Once you reference this include file at the top of your procedure, you can access any of the API shared variables. For more information about referencing include files see the *Progress Language Reference*. See the"Accessing and Editing the Current Query"section in this chapter for more information on how to use the API shared variables in a feature or integration procedure.

Using the Procedural Interface

Results provides an interface that allows you to manipulate Results queries and execute Results features from a feature or integration procedure. The procedure interface consists of six API procedures. All of the API procedures are located in the procedure library in the `$DLC/gui` directory. Table 3–3 provides summary information about the API procedures available for Results.

Table 3–3: API Procedures

API Procedure	Description
`aderes/sffire.p`	Executes a feature defined in Results from a feature or integration procedure.
`aderes/vgrpti.p`	Retrieves report information. This procedure returns information about the format of a report.
`aderes/vgtbll.p`	Retrieves the table list of the current query in Results.
`aderes/vgtbli.p`	Retrieves table information. This procedure accesses the join and selection criteria associated with a specified table in the current query in Results.
`aderes/vstbll.p`	Sets the table list for a new current query in Results.
`aderes/vsrpti.p`	Sets page size for the current query.
`aderes/vstbli.p`	Sets the table information associated with the current query.
`aderes/vsexpi.p`	Sets the export format for the current query.
`aderes/vslbli.p`	Sets the label format for the current query.

To use these API procedures in a feature procedure:

- Use the RUN statement to call an API procedure from a feature procedure.
- Define a local variable in the feature procedure for each input-output and output parameter specified in the API procedure call.

For more information, see the RUN Statement and DEFINE VARIABLE Statement in the *Progress Language Reference*.The following sections of this chapter describe how to use the different API procedures in a feature procedure.

Executing a Feature from a Feature Procedure

The API `sffire.p` procedure executes a feature from a feature procedure. With this capability, you can use features in Results as building blocks for other features. Use the following syntax to call the API `sffire.p` procedure from a feature procedure:

SYNTAX

```
RUN aderes/sffire.p
( INPUT featureId , INPUT args, OUTPUT result ) .
```

The following list describes the parameters associated with the API `sffire.p` procedure:

`INPUT` *`featureId`*

A string value representing the name of a feature as defined in Results. For information about Results features, see Appendix A, “Results Features.”

`INPUT` *`args`*

A string value representing one or more arguments passed to the specified feature. Specify multiple arguments as a comma-separated list. If the feature specified does not take arguments, pass the unknown value (?) for this parameter. For information about arguments for Results features, see Appendix A, “Results Features.”

`OUTPUT result`

A logical value that indicates whether the API procedure executed successfully.

If you use the `sffire.p` API procedure in a feature procedure, you must define a variable for the result output parameter, as shown in the following example:

```
  DEFINE VARIABLE result AS LOGICAL NO-UNDO.
                    .
                    .
                    .
RUN aderes/sffire.p (INPUT "FileOpen", INPUT ?, OUTPUT result).
```

In this example, FileOpen is a feature that does not take an argument list. If Results can execute the FileOpen feature, the OUTPUT parameter result returns a TRUE value.

Table 3–4 lists several Results features that you can use with the API `sffire.p` procedure. These features are important for developing other feature and integration procedures.

Table 3–4: Important Results Features

API Feature	Argument	Description
AdminProgInitialize	None	Clears all information from the API shared variables.
AdminProgInstantiate	None	Initializes any unset API shared variables with valid default values.
AdminProgWrite4GL	*proc-name,format*	Generates 4GL code for the current query and view, placing the source in the file specified with the *proc-name* argument. The *format* argument must be the character g or r. The character g generates stand-alone code. The character r generates code dependent on the Results application.

For information about additional features in Results, see Appendix A, "Results Features."

Testing the Results Version Number

The API shared variable qbf–vers contains a string value representing the Results product version number. The product version number consists of two digits separated by a decimal point, and followed by a letter (for example, 9.1A). Check the version number to verify that the environment you used to develop your feature and integration procedures matches the version of Results at the end-user site.

If the versions do not match, API changes between Results versions could cause erratic behavior. Feature and integration procedures that access API shared variables and procedures might yield error messages. Table 3–5 lists some of the error messages that can result from a change to the Results API between versions.

Table 3–5: The Results API and Error Messages

Error Message	Possible Problem
** "aderes/*proc-name*" was not found. (293)	The API procedure referenced in a feature or integration procedure is no longer supported in the current version of Results.
** Unknown Field or Variable name - *var-name*. (201)	The API shared variable referenced in a feature or integration procedure is no longer supported in the current version of Results.
Mismatched parameter types passed to procedure *proc-name*. (3230)	The parameters required by an API procedure are different in the current version of Results.
Procedure *proc-name* sent sub-procedure *proc-name* mismatched parameters.	The parameters required by an API procedure are different in the current version of Results.

These errors can prevent the feature or integration procedure from executing. To avoid this problem, rebuild your application. See Chapter 2, "Administering Results," for more information about rebuilding an application.

3.5.2 Accessing and Editing the Current Query

The following sections provide information about how to use the API shared variables and procedures to access and edit information associated with the current query in Results.

Determining the Current View

Use the qbf–module shared variable in your feature and integration procedures to determine and set the current view for the current query in Results. The qbf–module character variable holds one of the characters listed in Table 3–6.

Table 3–6: View Character Codes *(1 of 2)*

Value	Current View
?	No current query or view
b	Browse view
f	Form view

Table 3–6: View Character Codes *(2 of 2)*

Value	Current View
r	Report view
l	Label view
e	Data Export view

When the qbf–module variable contains the unknown value (?), there is no current query or view in the Results application window.

When you set the qbf–module variable in a feature procedure, make sure the output parameter controlling the redisplay of Results is set to TRUE. Consider the following feature procedure:

```
  DEFINE INPUT PARAMETER args AS CHARACTER NO-UNDO
  DEFINE OUTPUT PARAMETER winState AS LOGICAL NO-UNDO INITIAL FALSE.

{aderes/u-pvars.i}
qbf-module = ?.
winState = TRUE.
```

The last line of this feature procedure sets the winState output parameter to TRUE. This causes the Results window to redraw and display the changed module state when the feature procedure finishes executing.

Determining the Name of the Current Query

The shared variable `qbf-name` contains the name of the current query in Results. If you set the `qbf-name` variable in a feature procedure to a query that does not exist in the current query directory, Results creates a new temporary query definition. To save the new query definition permanently in the current query directory, execute the FileSave feature from a feature procedure or choose Query→ Save while the query is active in Results.

Accessing the Tables and Selection Criteria of the Current Query

Results provides API procedures that retrieve information about the tables in the current query.

The `vgtbll.p` API procedure retrieves a list of the tables referenced in the current query. Use the following syntax to call the `vgtbll.p` API procedure from a feature procedure:

```
RUN aderes/vgtbll.p (OUTPUT tableList).
```

The OUTPUT tableList parameter is a string value containing a comma-separated list of table or alias names referenced in the currently active query.

The vgtbli.p API procedure retrieves information about the join and selection criteria for a specified table referenced in the current query. Use the following syntax to call the vgtbli.p API procedure from a feature procedure:

```
RUN aderes/vgtbli.p (INPUT tableName, OUTPUT refTable, OUTPUT joinTable,
OUTPUT joinInfo, OUTPUT whereInfo, OUTPUT whereAsk, OUTPUT includeCode,
OUTPUT result).
```

The following list describes the parameters of the vgtbli.p API procedure:

INPUT *tableName*

A string value representing the name of a table or alias used in the current query.

OUTPUT refTable

If *tableName* is an alias, this parameter is a string value representing the name of the table associated with the alias. If *tableName* is an actual table name, this parameter returns the unknown value (?).

OUTPUT joinTable

A string value representing the name of a table to which the table specified with the first input parameter (*tableName*) is joined.

OUTPUT joinInfo

A string value containing the Progress 4GL statement relating the specified table with another table available in the currently active query.

OUTPUT whereInfo

A string value representing a selection expression in the WHERE clause of the query. This string value does not contain the keyword WHERE and contains only the selection criteria defined explicitly for the query. If this output parameter yields an empty string, there is no WHERE criteria for the corresponding table.

OUTPUT whereAsk

A string value representing the information used to prompt for a WHERE clause value at run time. This is the syntax for the string:

```
/*dataType,dbField,operator:promptString*/ TRUE
```

The database field, the data type of the database field, the operator of the criteria expression, and the prompt string are listed.

OUTPUT includeCode

A string value containing Progress 4GL statements to execute immediately after each record access from the specified table in the current query.

OUTPUT result

A logical value that indicates whether the API procedure executed successfully.

Determining the Query Sort Order

Use the qbf–sortby variable to retrieve and set the sorting information for the current query. The qbf–sortby variable contains BY clauses but does not contain the keyword BY. The list in the qbf–sortby variable has the following form:

SYNTAX

```
db-name.tbl-name.fld-name [ DESC ]
[ , db-name.tbl-name.fld-name [ DESC ] ] ...
```

The fields are positioned in the list based on sort precedence, with the primary sort field listed first. The letters DESC specified after a field name in the list indicate that the sort order for the field is descending. If the field is an array field, the field name is followed by a number in square brackets (for example, [#], where # is the array element selected for ordering).

Determining the Fields in the Current Query

Results places information about the fields of the current query into several different API shared variables. Many of the API shared variables that store information about the fields of the current query are arrays. In each of these field information arrays, there is one element per field in the current query. Results places information about each field into the field information arrays based on the order in which the fields are added to the current query. You can use a single index value to access information about a particular query field across these arrays. For example, an index value of 3 used with each field information array yields information about the third field added to the current query.

These are the shared variables and arrays that contain information about the fields of the current query:

- The qbf–rc# integer variable represents the total number of fields (database, calculated, and array elements) in the current query.

- The qbf–rcn variable is a character array containing names for each field in the current query. The contents of an element of this array vary depending on the information stored in the corresponding element of the qbf–rcc array. If an element in the qbf–rcc array is blank, the corresponding element in qbf–rcn is a field name. Otherwise, the element contains a comma-separated list of information about a calculated field.

  The format of the information list of a calculated field varies depending on the type of calculated field. The name of the calculated field is always the first entry in the list. By default, the calculated field name is qbf–*###*, where ### is a three-digit number representing the index of the current element in the qbf–rcn array. You can rename calculated fields using the Rename button in the Properties dialog box. The ENTRY(1,qbf–rcn[*n*]) function always returns the field name from an element of the qbf–rcn variable. Table 3–7 describes the format of the information list for each type of calculated field.

- The qbf–rcc variable is a character array used to store information for each calculated field and stacked array in the current query. (Stacked arrays display the elements in a column within a single field on the display.) If the first character is blank, the associated field is not a calculated field or a stacked array. For each calculated field in the qbf–rcc variable, the corresponding element in qbf–rcn contains information about the expression used to calculate the field. Table 3–7 describes the symbols used in an element of the qbf–rcc array to represent the different types of calculated fields.

Table 3–7: Calculated Field Symbols *(1 of 3)*

First Character	Description	Corresponding qbf–rcn Value Format
r	Running total.	*qFld, tFld* where *qFld* is the name of the query field and *tFld* is the name of the field used to calculate the running total.
p	Percent of total.	*qFld*, *pFld* where *qFld* is the name of the query field and *pFld* is the name of the field used to calculate the percent of total.
c	Counter. The c is followed by one of the following symbols: <, >, or *. The < symbol signifies that the count is based on the outer-most section of the report. The > symbol signifies that the count is based on the inner-most section of the report. The * symbol signifies that the count is based on all sections of the report.	*qFld*, *sVal*, *iVal* where *qFld* is the name of the query field, *sVal* is the starting value, and *iVal* is the incremental value used in the count operation.
e	Stacked array. The e is followed by the extent of the array (for example, e12 for customer.mnth–sales[ ]).	*qFld*, *aFld* where *qFld* is the name of the query field and *aFld* is the array field.w.

Table 3–7: Calculated Field Symbols *(2 of 3)*

First Character	Description	Corresponding qbf–rcn Value Format
s	String calculated field. The s is followed by a comma-separated list of the names of the fields used in the calculation.	*qFld*, *expr* where *qFld* is the name of the query field and *expr* is the expression used to calculate the field value.
n	Numeric calculated field. The n is followed by a comma-separated list of the names of the fields used in the calculation.	*qFld*, *expr* where *qFld* is the name of the query field and *expr* is the expression used to calculate the field value.
l	Logical calculated field. The l is followed by a comma-separated list of the names of the fields used in the calculation.	*qFld*, *expr* where *qFld* is the name of the query field and *expr* is the expression used to calculate the field value.

Table 3–7: Calculated Field Symbols *(3 of 3)*

First Character	Description	Corresponding qbf–rcn Value Format
d	Date calculated field. The d is followed by a comma-separated list of the names of the fields used in the calculation.	*qFld*, *expr* where *qFld* is the name of the query field and *expr* is the expression used to calculate the field value.
x	Lookup field.	*qFld*, *sTbl*, *sFld*, *mTbl*, *mFld*, *dFld*, *eStr* where *qFld* is the name of the query field, *sTbl* is the name of the source table, *sFld* is the name of the source field containing the lookup values, *mTbl* is the name of the match table, *mFld* is the name of the field in the match table containing values used for comparison in the lookup operation, *dFld* is the name of the field in the match table containing values to display for successful matches, and *eStr* is the error string to display if there is no valid match.

For calculated field types s, n, l, and d, ENTRY(1,qbf–rcn[*n*]) is the column name and SUBSTRING(qbf–rcn,INDEX(qbf–rcn[*n*],",") + 1) contains the expression. You cannot use ENTRY(2,...) to retrieve the expression, since it might have embedded commas.

- The qbf–rcg variable is a character array containing totals information for each field in the current query. Elements that contain an empty string have no totals information for the corresponding query field. If an element contains an ampersand (&), repeating values are hidden for the associated field.

  An element of the qbf–rcg variable can also contain one or more aggregate strings that consist of a leading letter followed by an integer or 0. An aggregate string must have one of the forms listed in Table 3–8.

Table 3–8: Aggregate String

Character	Description
tn	Total
cn	Count
an	Average
n*n*	Minimum
x*n*	Maximum

If *n* is 0, the total accumulation for the associated field is placed on the bottom line (summary line) of the report view of the current query. If *n* is an integer, it identifies an element in the qbf–sortby variable containing the name of a query field that triggers the accumulation.

An aggregate string can also contain a dollar sign ($), which indicates that the accumulate string is a totals-only string.

- The qbf–rcl variable is a character array containing labels for each field in the current query. Stacked column labels are supported. In stacked column labels, an exclamation point (!) breaks column labels into more than one level.
- The qbf–rcf variable is a character array containing format strings for each field in the current query. All Data Dictionary formats are supported.
- The qbf–rcp variable is a character array containing layout parameters related to the report, form, and label views for a field in the current query. Each element of the qbf–rcp variable has the following syntax:

```
rCol, rRow, fCol, fRow, lCol, lRow
```

These are the entries contained in each element of the `qbf-rcp` variable:

Variable	Description
rCol	An integer representing the column position in characters for the current field from the left side of the report view.
rRow	An integer representing the row position in lines for the current field from the top of the Report view.
fCol	An integer representing the column position in characters for the current field from the left side of the Form view.
fRow	An integer representing the row position in lines for the current field from the top of the Form view.
lCol	An integer representing the column position in characters for the current field from the left side of the Label view.
lRow	An integer representing the row position in lines for the current field from the top of the Label view.

Do not set an element of the qbf–rcp variable to the empty string. Instead, set the element to a character string containing five commas (“,,,,,”).

- The qbf–rcw variable is an integer array containing character widths for each query field in the report view. The width of each field includes format, label widths, and stacked labels. This variable is for information purposes only. Results calculates the character width of a field automatically when the query displays.

- The qbf–rct variable is an integer array containing a data type identifier for each field in the current query. The data type identifier is an integer value between 1 and 7 (inclusive) that maps to a list element contained in the qbf–dtype variable.

- The qbf–dtype character variable containing a comma-separated list of data types. The default value of this variable is a character array:

```
"character,date,logical,integer,decimal,raw,recid"
```

  Use this variable with the qbf–rct variable to determine the data type of a field in the current query.

Determining the Report Format of the Current Query

The vgrpti.p API procedure retrieves report format information associated with the current query. Use the following syntax to call the vgrpti.p API procedure from a feature procedure:

```
RUN aderes/vgrpti.p (OUTPUT leftOrigin, OUTPUT topOrigin,
OUTPUT columnSpace, OUTPUT lineSpace, OUTPUT pageLines,
OUTPUT headSpace, OUTPUT footSpace, OUTPUT pageEject,
OUTPUT sumReport).
```

The following list describes the parameters of the vgrpti.p API procedure:

OUTPUT leftOrigin

An integer representing left margin setting, in characters, from the output origin (page or display window).

OUTPUT topOrigin

An integer representing top margin setting, in characters, from the output origin (page or display window).

OUTPUT columnSpace

An integer representing the number of character units between fields on the report.

OUTPUT lineSpace

An integer representing the number of empty lines between text lines on the report.

OUTPUT pageLines

An integer representing the maximum number of lines allowed on a printed page of the report.

`OUTPUT headSpace`

An integer representing the number of lines between the report body and the report header.

`OUTPUT footSpace`

An integer representing the number of lines between the report body and the report footer.

`OUTPUT pageEject`

A string value containing the name of the report field used for page ejects.

`OUTPUT sumReport`

A logical value that specifies whether the report is a summary report.

The report view does not have to be active in Results in order to run the `vgrpti.p` API procedure.

Setting Table Information for the Current Query

The `vstbli.p` API procedure sets table information associated with the current query. Use the following syntax to call the `vstbli.p` API procedure from a feature procedure:

```
RUN aderes/vstbli.p (INPUT tableName, INPUT joinTable, INPUT joinInfo,  INPUT
whereInfo, INPUT whereAsk, INPUT includedCode, OUTPUT result).
```

The following list describes the parameters of the `vstbli.p` API procedure:

`INPUT tableName`

A string value representing the name of a table or alias used in the current query. The table name must be fully qualified (for example, db.table.name).

`INPUT joinTable`

A string value representing the name of the table to which tableName is joined, if any.

`INPUT joinInfo`

A string value containing the Progress 4GL code defining the join relationship between joinTable and tableName.

INPUT whereInfo

A string value representing a selection expression in the WHERE clause of the query. This string value does not contain the keyword WHERE and contains only the selection criteria defined explicitly for the query.

INPUT whereAsk

A string value representing the information used to prompt for a WHERE clause value at run time. This is the syntax for the string value:

```
/*dataType,Field,operator:promptString*/ TRUE
```

The field, the data type of the field, the operator of the criteria expression, and prompt string are listed.

INPUT includedCode

A string value containing Progress 4GL statements to execute immediately after each record access from the specified table in the current query.

OUTPUT result

A logical value that indicates whether the API procedure executed successfully.

Setting the List of Tables for the Current Query

The vstbll.p API procedure sets the list of tables associated with the current query. Use the following syntax to call the vstbll.p API procedure from a feature procedure:

```
RUN aderes/vstbll.p (INPUT tableList, OUTPUT result).
```

The following list describes the parameters of the vstbll.p API procedure:

INPUT tableList

A string value containing a list of tables to be used in the current query. The table names must be fully qualified (for example, db.table.name).

OUTPUT result

A logical value that indicates whether the API procedure executed successfully.

Setting the Export Format for the Current Query

The vsexpi.p API procedure sets the export format for the current query. Use the following syntax to call the vsexpi.p API procedure from a feature procedure:

```
RUN aderes/vsexpi.p (INPUT newFormat, OUTPUT result, OUTPUT oldFormat).
```

The following list describes the parameters of the vsexpi.p API procedure:

INPUT newFormat

A string value representing the name of the export format to make active for the current query. If the format name is not found in the list of existing formats, it is not set.

OUTPUT result

A logical value that indicates whether the API procedure executed successfully.

OUTPUT oldFormat

A string value representing the name of the old (prior to calling this API) export format. This information can be used to restore the export format to its original state at the end of the feature procedure.

Setting the Report Format for the Current Query

The vsrpti.p API procedure sets the report (page) format for the current query. Use the following syntax to call the vsrpti.p API procedure from a feature procedure:

```
RUN aderes/vsrpti.p (INPUT newFormat, OUTPUT result, OUTPUT oldFormat).
```

The following list describes the parameters of the vsrpti.p API procedure:

INPUT newFormat

A string value representing the name of the page format to make active for the current query. If the format name is not found in the list of existing formats, it is not set.

OUTPUT result

A logical value that signals whether or not the API procedure executed successfully.

OUTPUT oldFormat

A string value representing the name of the old (prior to calling this API) page format. This information can be used to restore the page format to its original state at the end of the feature procedure.

Setting the Label Format for the Current Query

The vslbli.p API procedure sets the label format for the current query. Use the following syntax to call the vslbli.p API procedure from a feature procedure:

```
RUN aderes/vslbli.p (INPUT newFormat, OUTPUT result, OUTPUT oldFormat).
```

The following list describes the parameters of the vslbli.p API procedure:

INPUT newFormat

A string value representing the name of the label format to make active for the current query. If the format name is not found in the list of existing formats, it is not set.

OUTPUT result

A logical value that indicates whether the API procedure executed successfully.

OUTPUT oldFormat

A string value representing the name of the old (prior to calling this API) label format. This information can be used to restore the label format to its original state at the end of the feature procedure.

3.5.3 Generating New Queries

You can generate new queries programmatically in Results from a feature or integration procedure. Query generation from a feature or integration procedure uses the vstbll.p API procedure and some special Results features.

The vstbll.p API procedure allows you to set the table list for a new query in Results. You cannot run the vstbll.p API procedure when there is a current query in Results. If there is an active query in Results, you must execute the AdminProgInitialize feature to clear the query prior to running the vstbll.p API procedure. Use the following syntax to call the vstbll.p API procedure from a feature procedure or integration procedure:

```
RUN aderes/vstbll.p (INPUT tableList, OUTPUT result).
```

The following list describes the parameters associated with the `vstbll.p` API procedure:

`INPUT` *`tableList`*

A string value containing a comma-separated list of table or alias names to establish the currently active query in Results. The tables specified must be available in the currently attached databases. If you intend to join a table to another table, you must specify both tables in the list in the order specified in the join. For example, the join Customer OF Order requires that the Customer table must be listed before the Order table in the table list.

`OUTPUT result`

A logical value that indicates whether the API procedure executed successfully.

To generate queries from a feature or integration procedure, follow these general steps:

1 ♦ Execute the AdminProgInitialize feature using the `sffire.p` API procedure to clear the current query from Results.

2 ♦ Run the `vstbll.p` API procedure to set the table list for the new query.

3 ♦ Define fields and format information for the new query using the API shared variables and procedures. At a minimum, set the qbf–module and qbf–name variables and define a field for the query.

4 ♦ Execute the AdminProgInstantiate feature using the `sffire.p` API procedure to initialize any unset API shared variables with valid default values. You must execute this feature before you attempt to display or run the new query.

5 ♦ Save the new query, generate 4GL code for the query, then print it or display it in Results.

The following feature procedure generates a new query, then displays it in Results:

The commented numbers in the code refer to these notes:

tstFeat.p

```
/* 1 */

{ aderes/u-pvars.i }
/* 2 */
DEFINE INPUT  PARjAMETER args AS CHARACTER NO-UNDO.
DEFINE OUTPUT PARAMETER winState AS LOGICAL NO-UNDO.
/* 3 */
DEFINE VARIABLE result AS LOGICAL NO-UNDO.
/* 4 */
RUN aderes/sffire.p("AdminProgInitialize", ?, OUTPUT result).
/* 5 */
RUN aderes/vstbll.p("sports.Customer,sports.Order", OUTPUT result).
    IF result = TRUE THEN DO:
/* 6 */
   RUN aderes/vstbli.p(INPUT "sports.Customer", INPUT ?, INPUT ?,
    INPUT "Customer.State = ""MA""", INPUT ?, INPUT ?, OUTPUT result).
   RUN aderes/vstbli.p(INPUT "sports.Order", INPUT "sports.Customer",
    INPUT "OF sports.Customer", INPUT ?, INPUT ?, INPUT ?,
    OUTPUT result).
/* 7 */
    ASSIGN
    qbf-module = "r"
    qbf-name = "newque"
    qbf-rc# = 2
    qbf-rcn[1] = "sports.Customer.Name"
    qbf-rcl[1] = "Cust Name"
    qbf-rcf[1] = "X(25)"
    qbf-rct[1] = 1
    qbf-rcp[1] = ",,,,,,,"
    qbf-rcn[2] = "sports.Order.Order-num"
    qbf-rcl[2] = "Ord Num"
    qbf-rcf[2] = ">>>>9"
    qbf-rct[2] = 4
    qbf-rcp[2] = ",,,,,,,".
END. /* DO */
/* 8 */
RUN aderes/sffire.p(INPUT "AdminProgInstantiate", INPUT? ,

    OUTPUT result).
/* 9 */
    winState = TRUE.
```

1. Includes the `u-pvars.i` file to define the API shared variables for access in the procedure.

2. Defines the input and output parameters required by a feature procedure.

3. Defines a variable to capture the output values of the different API procedures used in this feature procedure.

4. Executes the AdminProgInitialize feature using the `sffire.p` API procedure to clear the current query from Results.

5. Sets the table list for the new query. The example feature procedure uses tables from the sports database.

6. Defines the join and selection criteria for the tables specified for the new query.

7. Defines two fields for the new query.

8. Instantiates the rest of the data structures of the new query in preparation for use in Results.

9. Sets the output parameter of the feature procedure to TRUE, which tells Results to redraw the window and display the new query defined in the feature procedure.

For more information about the language elements used in the this example procedure, see the *Progress Language Reference*.

3.6 Output Solutions in Results

By default, Results allows the user to view query output on screen using the Print Preview option and to direct output to the default system printer, a file, or the system clipboard using the Print menu options. You can develop your own output solutions and integrate them as features in Results. Several sample output feature procedures located in the `aderes` product directory provide examples for your own output solutions. Table 3–9 describes the sample output feature procedures.

Table 3–9: Sample Output Feature Procedures

Output Procedure	Description	Syntax of Required Input Parameter String
u-cprint.p	Sends the output of the current query to a specified printer with control characters using the OUTPUT TO and PUT CONTROL statements.	*printer* [, *control*]... where *printer* is the operating system name of a printer and *control* is an integer representing the ASCII character value of a printer control character.
u-thru.p	Sends the output of the current query to a specified printer using the OUTPUT THROUGH statement. This procedure is designed exclusively for use on character-mode systems.	*printer* where *printer* is the operating system name of a spooled print device.
u-to.p	Sends the output of the current query to a specified printer using the OUTPUT TO statement.	*printer* where *printer* is the name of a printer.
u-pprog.p	Generates a temporary procedure file for the current query, then runs a specified Progress procedure with the generated procedure as an argument.	*procedure* where *procedure* is the name of a Progress procedure.
u-pview.p	Sends the output of the current query to a temporary text file, then runs a specified Progress procedure with the temporary text file as an argument.	*procedure* where *procedure* is the name of a Progress procedure.

NOTE: You cannot use these sample output procedures to direct output for queries in the browse and form views.

Follow these steps to add a sample output procedure to Results as a feature and menu option:

1 ♦ Choose Query→ Customize→ Feature to access the Feature Editor.

2 ♦ Define the feature, then choose Add. For example, the following list provides information that defines a feature that uses the `u-cprint.p` procedure to direct query output and printer control characters to a specified printer:

- **Name** — PrintCompressed
- **Program** — /usr/dlc/gui/aderes/u–cprint.p
- **Argument** — lpr,15
- **Label** — To &lpr compressed

3 ♦ Choose Query→ Customize→ Menu Layout to access the Menu Editor.

4 ♦ Add the new feature to the Query→ Print menu.

For more information about output routing solutions in Progress, see the *Progress Programming Handbook*.

3.7 Export Solutions in Results

Results provides a robust set of standard export formats, but you can also develop your own standard export format. There are two basic techniques for implementing a standard export format for use at a user site:

- Edit the existing Sample Export format.
- Edit the `results.l` file to define a new export format.

This section describes the Sample Export Format. For information about the contents of the `results.l` file, see Appendix C, "Results Files."

The Sample Export format is one of the default standard export formats and performs an export using the default Progress export format. The Progress source procedure that implements the Sample Export format is `u-export.p` and is located in the `aderes` product directory.

You can edit the `u-export.p` procedure and code your own export solution. Note that the `u-export.p` procedure executes once for each field defined for export in the query. These fields are listed in the qbf–rcn API shared variable for the active query. The `u-export.p` procedure defines several input parameters that pass information about the current export field from

Results to the procedure. Descriptions for each of these input parameters are located in the comment text of the `u-export.p` procedure.

The body of the `u-export.p` procedure builds an EXPORT statement using the input parameter values. The `u-export.p` procedure uses the PUT statement to write the EXPORT statement to a temporary file that Results executes to export data from the query. For more information about the EXPORT and PUT statements, see the *Progress Language Reference*.

If you copy and edit the `u-export.p` file, make sure that the new procedure is located in a directory that is listed in the `PROPATH` before the Progress product directory (`DLC`).

3.8 Results Help System

The Results help system contains complete information about the Results product. Progress supplies the source files for the Results help system so that programmers can integrate help information for extensions to Results into the Results help system and distribute a single help file. You will find these Help source files in the directory `%DLC%\src\prohelp`.

Table 3–10 lists the source files of the Results help system.

Table 3–10: Source Files for the Results Help System *(1 of 2)*

Source File	Description
`reseng.hpj`	The help project file for the Results help system. This ASCII file contains the instructions that the help compiler uses to build the Results help system. Do not change the existing information in the MAP section of this file. If you change this information, you will break the Results help system.

Table 3–10: Source Files for the Results Help System *(2 of 2)*

Source File	Description
reseng.rtf reseng2.rtf	The help topic files for the Results help system, in rich text format (RTF). These files contains the text of the Results Help System. You can edit the contents of these files and include additional information to document extensions to Results. Use a word processor that supports the RTF standard or a text editor to edit these files.
header.shg results.shg btnbar2.shg first.shg hdr-res.shg last.shg next.shg previous.shg vcr-clse.shg vcr-frst.shg vcr-last.shg vcr-next.shg vcr-prev.shg vcr-prnt.shg browse-u.bmp export-u.bmp form-u.bmp labels-u.bmp open-u.bmp prevw-u.bmp previous.bmp print-u.bmp rpt-u.bmp save-uib.bmp select-u.bmp sort-u.bmp vcr-clse.bmp vcr-frst.bmp vcr-last.bmp vcr-next.bmp vcr-prev.bmp vcr-prnt.bmp window.bmp	Graphics files referenced in the reseng.rtf help topic file and used to build the Results help system. The files are of two kinds: MS-Windows Bitmap (.bmp) and Segmented Hypergraphics (SHED) bitmaps (.shg). If you use SHED format graphics, you can lock the resolution so that Windows's Help system does not attempt to optimize the bitmap resolution to suit the particular monitor in use. You can substitute your own graphics files for those (for example, header.shg) that are specific to the Results product name and identity.

For more information about how to author, build, and integrate help systems into a Progress application, see the *Progress Help Development Guide*.

3.9 Results Main Application Procedure

Most of the discussion in this chapter focuses on calling other Progress procedures from Results. But you can also integrate Results into an existing Progress application. The main procedure for the Results application is `results.p`, located in the Progress product directory (`DLC`). To run Results from another Progress procedure, use the following statement:

```
RUN results.p.
```

If you run Results from a Progress application that has its own application window, use the technique outlined below to manage user focus and multiple windows:

```
/* Template Calling RESULTS from a Progress application */

/* 1 */
   DEFINE VARIABLE parentWin AS WIDGET-HANDLE NO-UNDO.

/* 2 */
   ASSIGN

    parentWin = CURRENT-WINDOW
    parentWin:WINDOW-STATE = WINDOW-MINIMIZED
    parentWin:SENSITIVE = FALSE.

/* 3 */

   RUN results.p.


   ASSIGN

/* 4 */
   CURRENT-WINDOW = parentWin

/* 5 */
    parentWin:WINDOW-STATE = WINDOW-NORMAL
    parentWin:SENSITIVE = TRUE.
```

The commented numbers refer to the notes:

1. Create a variable to hold the widget handle of the parent application window.

2. Place the widget handle of the parent application window into the new variable. The CURRENT–WINDOW session handle returns the widget handle of the parent application window.

3. Minimize the parent application window and disable it for user input while the procedure displays the Results application window.

4. Restore the CURRENT–WINDOW handle to the original window handle.

5. Restore the parent application window and enable it for user input after the user closes the Results application window.

For more information about general programming concepts and techniques, see the *Progress Programming Handbook*. See the *Progress Language Reference* for information about language elements in the Progress 4GL.

3.10 Code Generation

You can use Results to generate a Progress 4GL procedure for a query view and use the procedure in other Progress applications. Results provides two ways to generate code:

- From within Results using the Query→ Generate menu option
- From an application using the AdminProgWrite4GL feature

Regardless of which way you generate the code, you must consider how to prompt users for input. See the "Ask at Run Time Values" section for information about prompting users.

3.10.1 Generating Code from Within Results

The Query→ Generate menu option generates a stand-alone Progress procedure for the current query and view. Table 3–11 provides some general information about how Query→ Generate generates code for the current query and view.

Table 3–11: Code Generation Using Query→ Generate *(1 of 2)*

View	Code Generation Information
Browse	For interactive use only. The code contains a WAIT–FOR statement but does not explicitly create a window widget.
Form	For interactive use only. The code contains a WAIT–FOR statement but does not explicitly create a window widget.
Report	The DISPLAY statement is used to output data. The OUTPUT statement is used to output the report to TERMINAL PAGED.

Table 3–11: Code Generation Using Query→ Generate *(2 of 2)*

View	Code Generation Information
Label	For noninteractive use only. The PUT statement is used to output data to `label.dat`.
Export	For noninteractive use only. The EXPORT statement is used to output data to `export.dat`.

NOTE: Code generated for the form view of a query does not contain the Query By Example button or functionality.

3.10.2 Generating Code from an Application

You can also generate code by executing the AdminProgWrite4GL feature in a feature procedure. The AdminProgWrite4GL feature generates Progress 4GL code for the report, label, or export view of a query. However, the AdminProgWrite4GL feature does not generate code for the browse and form views of a query.

Arguments to the AdminProgWrite4GL feature allow you to specify an output file for the code and to determine whether the generated code is stand-alone Progress code or dependent on Results. The following statement shows the syntax required to execute the AdminProgWrite4GL feature in a feature procedure:

```
RUN aderes/sffire.p(INPUT "AdminProgWrite4GL", INPUT "proc-name, format",
OUTPUT result).
```

The *proc-name* argument specifies the name of the operating system file where Results stores the generated code. The *format* argument must be the character g or r. The character g generates stand-alone code. See Table 3–11 for information about the stand-alone code that the AdminProgWrite4GL feature generates for different types of views. The character r generates code that is dependent on the Results application. Use the Results-dependent format to generate and run queries in Results that produce data in preparation for use with other applications that are integrated into the Results menu system (for example, business graphics applications).

3.10.3 Ask at Run Time Values

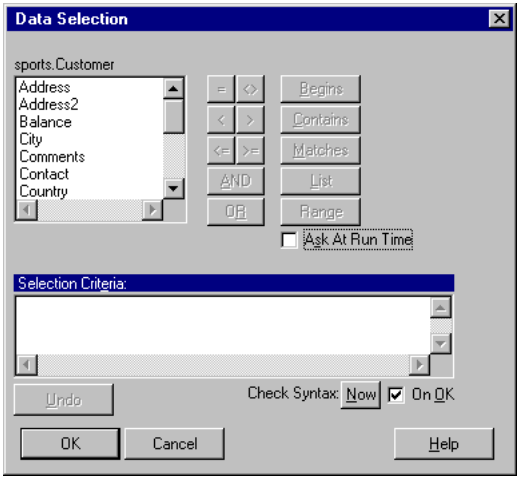
You can prompt users for input in generated code with the Ask At Run Time feature. There are two parts to the process. First, you specify that you want Results to accept input at run time. Then, when you generate the code, you can either specify the input values yourself or you can have Results prompt users for the values when they run the application containing the code.

If you choose to prompt users for the value when the code is run, the generated code includes an UPDATE statement for the Ask At Run Time phrase.

Specifying the User Prompt

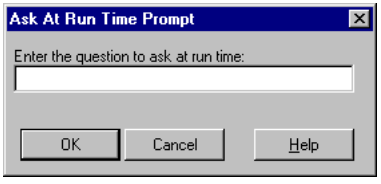
You specify the user prompt when you are creating the query. Follow these steps to specify the user prompt:

1 ♦ Choose Data→ Selection. If the query includes multiple tables, the Available Tables dialog box appears. Select the desired table, then choose OK. The Data Selection dialog box appears. If the query contains a single table, the Data Selection dialog box appears immediately:



2 ♦ Activate the Ask At Run Time toggle box.

3 ♦ Specify the selection criteria. The Ask At Run Time Prompt dialog box appears after you select an operator:



4 ♦ Enter the string you want Results to use when it prompts the user for input, then choose OK. The Data Selection dialog box reappears. Notice that the selection criteria now contains the run-time expression, including the string you just entered.

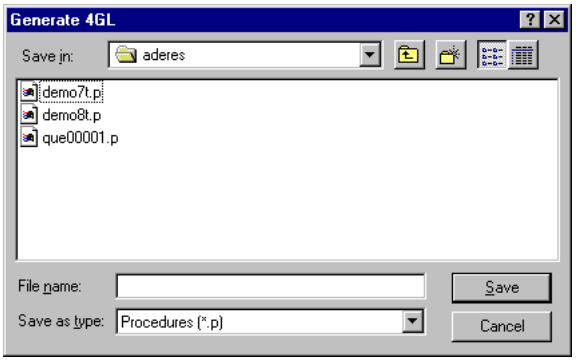
5 ♦ Choose OK to save the specified criteria and return to the main window.

Now, when you run the query, Results prompts the user for the selection criteria.

Generating the Query

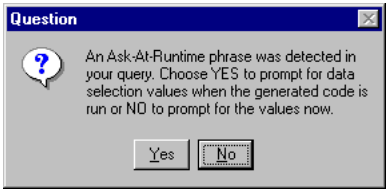
Follow these steps to generate the query:

1 ♦ Choose Query→ Generate. The Generate 4GL dialog box appears:



2 ♦ Specify the name of the file, then save the file. If the file already exists, Results prompts you to replace the file.

After you specify the file name, Results generates the code. Because you created a prompt phrase, the following dialog box appears:



3 ♦ Choose Yes if you want to prompt the user for the input values or No if you want to specify the input values now. If you choose Yes, Results prompts the user for input when the code is run.

For example, if you specify to ask the user “What is the customer’s number?”, the following dialog box appears when the code is run:



4

Deploying Results Applications

This chapter describes how to deploy a Results application to an end-user environment. It contains information on the following topics:

- The deployment process and site requirements
- Application files for deployment
- Application databases
- The application directory structure
- The Results help system
- Application compilation
- The fastload system

The information in this chapter is for Progress programmers who are familiar with Results and understand general deployment issues for Progress applications. This chapter describes deployment issues specific to Results and provides references to other Progress Software Corporation publications for general deployment information.

4.1 Deployment Process and Site Requirements

Deployment is the process of distributing an application for use in an end-user environment. Before you can deploy a Results application to an end-user environment, you must confirm that the environment has a licensed version of Results installed and also has Progress compilation capabilities.

There are a number of steps in the deployment process that you must perform in the development and end-user environments. Follow these steps to begin the deployment process for a Results application in the development environment:

1. Identify the files and directory structure of your Results application.
2. If you must deploy all or a portion of your application databases, dump the appropriate data and data definitions from your databases.
3. Transfer the application and database files to the distribution media.

Once you have the application and database files ready for distribution, the deployment process shifts to building the application in the end-user environment. To build a Results application in the user environment, do the following:

1. Create the application directory and file structure.
2. If you deployed all or a portion of your application databases, create the application database if necessary and then load the deployed data definitions and data.
3. If you edited the source files for the Results help system, compile the help system and move the resulting help file into the `prohelp` product directory.
4. Compile the deployed feature and integration procedures using the Application Compiler.
5. Start Results and rebuild the fastload system.

The remaining sections provide more information about the deployment process and the application components involved in this process.

4.2 Application Files for Deployment

When you deploy a Results application, you are deploying only the files that are unique to your application. You are not deploying the entire Results product. To deploy the entire Results product would be a violation of the Results product license, thus a licensed version of Results must exist in the end-user environment.

As a general rule, deploy only the source files and not the r-code files of a Results application. Differences between the development and user environments and possible changes in the application directory structure might require that you recompile your application files in the end-user environment. Results must also recompile the fastload system for your application in the end-user environment. The end-user environment must have Progress compilation capabilities.

The following sections provide information about the files that you should deploy for a Results application.

4.2.1 The Deployment Report

As you build a Results application, Results creates and maintains a deployment report. The deployment report lists all of the databases, procedure files, ASCII files, and graphics files that are part of your application in Results. These are the files that you must deploy to install your application in end-user environments.

The name of the deployment report consists of the first five characters of the current QC7 file followed by .txt. Results creates the deployment report in the same directory as the QC7 file. The following procedure is a sample deployment report for a simple Results application:

sport.txt

```
# Progress RESULTS Deployed Application FileList
# This reports does not provide Public Query Directory Information

baseResourceFile= aderes/results.l
configurationFile:ascii= sports.qc7
configurationFileAltSrc:featureFile= sportf.p
configurationFileAltSrc:menuFile= sportmt.p
database:1= sports
defaultIcon:Graph= C:/PROSPRT/GRAPH-D.BMP
defaultIcon:Graph= C:/PROSPRT/GRAPH-I.BMP
defaultIcon:Graph= C:/PROSPRT/GRAPH-U.BMP
featureFileSrc:Cmprint= C:\PROSPRT\CMPRINT.P
featureFileSrc:Graph= C:\PROSPRT\PRSGRAPH.P
integrationPointSrc:define= C:\PROSPRT\PRSDEF.P
integrationPointSrc:direct= C:\PROSPRT\PRSDIR.P
integrationPointSrc:field= C:\PROSPRT\PRSFLD.P
integrationPointSrc:login= C:\PROSPRT\PRSLOGN.P
integrationPointSrc:logo= C:\PROSPRT\PRSLOGO.P
integrationPointSrc:menu= C:\PROSPRT\PRSSEC.P
integrationPointSrc:sens= C:\PROSPRT\PRSMNU.P
integrationPointSrc:table= C:\PROSPRT\PRSTBL.P
integrationPointSrc:where= C:\PROSPRT\PRSSEL.P
logoIcon= C:/PROSPRT/PROSPRT.ico
```

Each file entry in the deployment report has a label that identifies how the file is integrated into Results. The following list describes the deployment report labels:

`baseResourceFile`

The relative pathname of the `results.l` file. Deploy this file only if you altered it to define export formats, page sizes, and label formats.

`configurationFile:asci`

The pathname of the QC7 file.

`configurationFileAltSrc:featureFile`

The pathname of the feature definition file. This file loads the feature definitions for your application into Results. Results generates a compiled, optimized version of this file for the fastload system. You can set the name of this file with the Query→ Customize→ Deployment menu option.

`configurationFileAltSrc:menuFile`

The pathname of the menu definition file. This file loads the menu system and tool bar configuration for your application into Results. Results generates a compiled, optimized version of this file for the fastload system.

`database:`*`n`*

The name of a database referenced in the query configuration file. The deployment report contains an entry for each database referenced in the query configuration file. The *n* represents the number of the database in the order of listing in the query configuration file. In most cases, the application databases already exist in the user environment.

`defaultIcon:`*`featureName`*

The pathname of an icon specified for a programmer-defined feature. The *featureName* parameter is the name of the feature associated with the icon. The deployment report typically has three defaultIcon entries for each new feature to identify graphics files associated with the three states of the icon: up, down, and insensitive. The following excerpt from a deployment report shows the graphics files associated with a feature called Graph:

```
defaultIcon:Graph= C:/PROSPRT/GRAPH-D.BMP
defaultIcon:Graph= C:/PROSPRT/GRAPH-I.BMP
defaultIcon:Graph= C:/PROSPRT/GRAPH-U.BMP
```

`featureFileSrc:`*`featureName`*

The pathname of a source procedure file associated with a programmer-defined feature. The *featureName* parameter is the name of the new feature. The following excerpt from a deployment report shows the procedure file associated with a feature called Cmprint:

```
featureFileSrc:Cmprint= C:\PROSPRT\CMPRINT.P
```

The deployment report contains a featureFileSrc entry for each programmer-defined feature.

`integrationPointSrc:`*`intPoint`*

The pathname of a source procedure file associated with an integration point. The *intPoint* parameter is an identifier for an integration point. These are the valid identifiers and the names of the corresponding integration points:

- **define** — Initialize Vars and Interface
- **direct** — Query Directory Switch Program
- **field** — Field Security
- **login** — Login Program
- **logo** — Logo Screen
- **menu** — Feature Security
- **sens** — Menu Item Override
- **table** — Table Security
- **where** — Table Selection Security

The deployment report contains an integrationPointSrc entry for each integration point that has an assigned integration procedure.

`logoIcon`

The pathname of a graphics file defined as the application icon for Results in the window system.

The following section lists additional files that do not appear in the deployment report, but are potential candidates for deployment as part of a Results application.

4.2.2 Other Possible Deployment Files

The deployment report lists the majority, but not all, of the files that must be deployed with a Results application. There are several files in a Results application that the product does not track or manage:

`u-export.p`

The procedure file associated with the Sample Export format in the Standard Export list. If you implement a custom export solution using this file, you must deploy the `u-export.p` file.

Application files

ASCII files, graphics files, include files, and other procedure files referenced in feature and integration procedure files. These files do not appear in the deployment report. You must track and manage these application files.

Help Files

The source files for the Results help system. If you edited the help source files for the Results help system, you must deploy the help source files and build the help system in the user environment.

Query Directories

Public and user query directory files containing predefined queries. If you deploy a query directory, you must also deploy the source files associated with the queries defined in the query directory. Results generates a source file for a query containing information that defines the query. The name of each source file begins with characters “que” followed by a unique five-digit number and the file extension.p (`que#####.p`).

Note the directory locations of these additional deployment files so that you can reimplement this application directory structure in the user environment.

4.2.3 Graphics Files

When deploying a Results application from one window system to another, you must convert your graphics files to a format that Progress supports for the target windowing system. For Windows, Progress supports the use of bitmaps (BMP), icons (ICO), and cursors (CUR) in your applications.

4.3 Application Databases

To deploy all or a portion of an application database, you must perform a dump-and-load operation. These are the situations that force you to deploy all or a portion of an application database from the development environment to the user environment:

- If the application database does not exist in the user environment, you must deploy the entire database.
- The application database exists in the user environment, but there are some changes to the schema of the database that you must deploy to the user environment.
- The application database exists in the user environment, but there is new data (for example, new user definitions in the _User file) that you must deploy to the user environment.

A dump operation produces text files containing data definitions (.df) files and data (.d) files. Transfer these data definition and data files to the user environment to recreate the application database or to load updates into an existing application database. When you deploy a database, remember to deploy the source code for any trigger procedures associated with the application database.

For more information about database triggers, see the *Progress Programming Handbook*. For more information about dumping and loading databases, see the *Progress Database Administration Guide and Reference*.

4.4 The Application Directory Structure

As a general rule, the directory structure of your Results application should be similar in the development and user environments. The directory structure of a Results application is reflected in the pathnames contained in the deployment report for the application.

NOTE: In Results, all dialog boxes that allow you to browse through directories and select files in your operating system return absolute pathnames for the selected files. In most circumstances, you can specify a relative pathname for a file by typing a pathname in an input field. For example, the Files button in the Feature Editor dialog box allows you to specify an absolute pathname for a feature procedure. You can enter a relative pathname for a feature procedure by typing the pathname in the Program field.

Relative pathnames and the PROPATH variable provide some flexibility in the implementation of an application directory in the user environment. The PROPATH is a list of directory pathnames that Progress uses to locate directories and files at run time. For example, Results uses the PROPATH to locate a QC7 and QD7 files. If you use relative pathnames and your application files are not located in the current working directory or a Progress product directory, you must include the directory where the files are located in the PROPATH. For Windows environments, the PROPATH environment variable is specified in the [Startup] section of the environment (.ini) file.

For more information about the PROPATH environment variable file, see the *Progress Database Administration Guide and Reference* and the *Progress Installation and Configuration Guide Version 9 for UNIX* or the *Progress Installation and Configuration Guide Version 9 for Windows*.

If you create a directory structure in the user environment that is different from the one used to develop the application, you must edit the QC7 file, the feature file (*base*`f.p`), and the UI file (*base*`mt.p`) of the application. These files contain pathnames that must be changed to reflect the new application directory structure in the user environment. You must edit these files before starting Results in the user environment. See Appendix C, "Results Files," for more information about QC7 files, feature files, and user interface (UI) files.

4.5 The Results Help System

If you edited the source files for the Results help file, you must deploy the source files to the user environment and use them to rebuild the help file. These are the basic steps for building a help file:

1. At the operating system prompt, change to the directory containing the help project file (`reseng.hpj`) for the Results help system.

2. Use the PROHC utility to build the application help file:

```
prohc reseng.hpj
```

   The PROHC utility is located in the `DLC/bin` directory.

3. Move the new application help file (HLP) to the `DLC/prohelp` directory. The application help file must have the name `reseng.hlp`.

4. Clean up the help source files. You can delete them or move them to a safe place for subsequent help compilations.

For more information about building help systems, see the *Progress Help Development Guide*.

4.6 Application Compilation

Use the Application Compiler or the COMPILE statement in the user environment to compile the feature, integration, and database trigger procedures of your application and your application database. You must be connected to the application database to compile procedures that access the databases. For information about the Application Compiler, see the *Progress Basic Development Tools* online help. For information about the COMPILE statement, see the *Progress Language Reference*.

4.7 The Fastload System

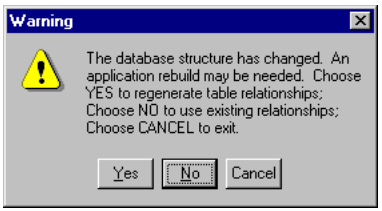
The *fastload system* is the set of r-code files that optimizes the configuration of Results at startup. Results automatically generates and updates the fastload system associated with the current QC7 file.

NOTE: A fastload system for a Results application should not exist in the user environment prior to the initial startup of the application. Do not distribute the fastload system for an application from the development environment to the user environment. The fastload system for a Results application in the user environment must be generated as part of the initial startup of the application and must reflect the application directory structure implemented in the user environment.

Follow these steps to start and generate a fastload system for a Results application in an end-user environment:

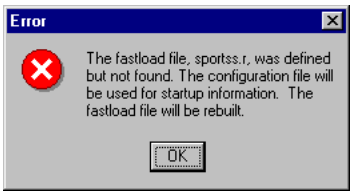
1 ♦ Start up Results. The order of the application databases specified in the startup command must be in the same order as used to generate the QC7 file in the development environment. For information about startup commands for Results, see Chapter 1, "Results for Administrators and Programmers."

2 ♦ At startup, Results checks the QC7 file for information about the required application databases. The QC7 file contains an integer, called a checksum number, for each application database from the development environment. The checksum number for a database is the sum of the last schema modification date for each table in the database. Results compares the checksum numbers from the development environment against the modification dates of the application databases in the user environment. If there is a difference, then the following warning message appears:



Choose No to continue with the initial startup without rebuilding the table relationships.

3 ♦ After analyzing the application databases, Results checks for the existence of a fastload system. If it cannot find one, it displays the following error message:



When you dismiss this error message, Results uses the QC7 to configure the features, menu system, and toolbar. This initial configuration process might take awhile because Results cannot use a compiled fastload system.

4 ♦ Exit Results. Before shutdown, Results updates the QC7 file with new checksum numbers for the application databases and regenerates the fastload system for the installation.

Your Results application is ready to use.

5

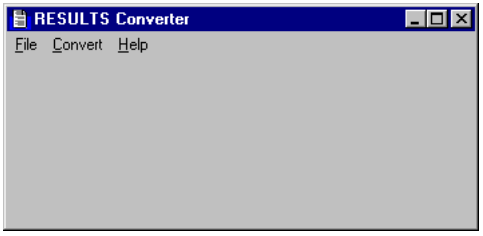
Converting Character-based Results Applications to Graphical Results

This chapter describes how to use the Results Converter to convert character-based Results applications to graphical Results. It contains information on the following topics:

- Results Converter basics
- Converting applications

5.1 Results Converter Basics

The Results Converter converts Results Version 6 and Version 7 character-based applications to Results Version 8 graphical applications. The Converter allows you to convert the configuration file, the query directory files, and all associated query, report, label, and export files, or all files. When you start the Converter, it prompts you to specify which type of conversion you want to perform. After you finish the conversion, the Converter main window appears:



Because you are converting files from one type of environment to another, not all the files convert with a one-to-one correspondence. The following sections describe how the Converter converts the files:

- Converted information
- Unconverted information
- Additional graphical information

5.1.1 Converted Information

This section describes the information that the Converter converts from character-based to graphical format. The Converter converts the following information:

- File conversions
- Printer information
- Auto-selection of label fields
- Module and query function permissions
- Report layout

- Relationships
- View files
- Queries

NOTE: Before converting your files, note that the graphical Results procedure library (`aderes.pl`) and the Converter r-code (`$DLC/gui`) must be in your `PROPATH`.

File Conversions

When the Converter converts Progress Version 6 or Version 7 character-based files, it converts them to the closest Progress Version 9 graphical equivalent.

Table 5–1 lists the original character-based files and their graphical equivalents.

Table 5–1: File Conversions *(1 of 2)*

Character-based File	Description	Graphical File	Description
table-name`.f` *table-name*`.i` *table-name*`.p`	Files generated when the application is built. There is a set of three files for each table in the database.	A form view `QUE0000n.p` is created for each set of files per table name. They are added to the `public.qd7` directory.	N/A
database`.QC`	Configuration file	*database*`.QC7`	Configuration file
username`.QD`	Query directory that contains the list of queries created by *username*.	*username*`.QD7`	Query directory that contains the list of queries created by *username*.

Table 5–1: File Conversions *(2 of 2)*

Character-based File	Description	Graphical File	Description
EXP0000*n*.p	Export view queries	QUE0000*n*.p	No distinction in name for queries in different views. The query type is now part of the file information.
LBL0000*n*.p	Label view queries		
REP0000*n*.p	Report view queries		
RES0000*n*.r	Fastload files that contain application information in compiled form that speeds up load time.	*bases*.r *baser*.r *base*00*n*.r *base*f.r *base*mt.r	Fastload files. By default, the Fastload name is the first five characters of the logical database name with additional characters to indicate the specific function.

Printer Information

For each printer definition, the Converter creates a feature and adds it to the Results Query→ Print menu. The printer type dictates how the information is converted. Table 5–2 lists the printer conversion information.

Table 5–2: Printer Conversion *(1 of 2)*

Printer Type	Character-based Description	Graphical Conversion
TERMINAL	Print Preview.	No conversion, Print→ Preview.
FILE	Print to a file.	No conversion, Print→ File.
PRINTER	Print to default printer.	No conversion, Print→ Printer.
term	TERMINAL, Output to Terminal Paged.	No conversion, Print→ Preview.
to	TO, Output to Printer/File.	Create feature, connect to `u-to.p` or `u-cto.p`.

Table 5–2: Printer Conversion *(2 of 2)*

Printer Type	Character-based Description	Graphical Conversion
thru	THROUGH a spooler or filter.	No conversion because this definition is not supported in Windows.
view	Send the report to a file and execute the application's 4GL.	Create feature, connect to `u-pview.p`.
page	To a screen with next-prev page support.	No conversion, Print→ Preview.
prog	Send procedure file to application's 4GL program to start/end output stream.	Create feature, connect to `u-pprog.p`.
file	Ask the user for filename, then send the output to the file.	No conversion, Print→ File.

Auto Selection of Label Fields

The label field information in the configuration file converts without any changes.

Module and Query Function Permissions

Module permission provides site security for each module. Query function permissions provide security for each type of query. The Converter converts both types of permissions and adds the information to the security portion of the graphical configuration file. When there is no corresponding graphical feature for a character-based feature, the Converter does not convert the character-based feature. Table 5–3 lists the query feature conversions.

Table 5–3: Query Feature Conversions *(1 of 2)*

Character-based Feature	Graphical Feature
query-add	Record Add
query-browse	Browse View
query-copy	Copies a record
query-delete	Record Delete

Table 5–3: Query Feature Conversions *(2 of 2)*

Character-based Feature	Graphical Feature
query-exit	Exit
query-first	N/A
query-help	Help menu
query-info	N/A
query-join	Joins to related records
query-last	N/A
query-module	Each type of View
query-next	N/A
query-order	Switches to a different sort index
query-prev	N/A
query-query	QBE for a form
query-total	Calculates the total number of records in the current set
query-update	Record Update
query-user	N/A
query-view	File Open
query-where	Selection

Report Layout

The report layout portion of the configuration file sets the default layout for a report. The converter converts all of this information, except for the header and footer information.

Relationships

The relationships portion of the configuration file identifies the relationships or joins for an application. All of this information converts to the graphical version of the configuration file.

View Files

The graphical and character-based versions of Results use the same basic techniques for saving query information. Both versions include a comment holding the query information at the beginning of a form, label, or export file. Both versions read this section to lay out the view in the user interface.

However, while both version have these similarities, the following differences exist between the two versions:

- Character-based query files include character-based 4GL. The Converter does not convert this code, but instead uses this information to create an equivalent query in the graphical 4GL.
- Graphical queries are defined only once. Each view type creates the user interface for the query. In contrast, if users want three views of a query in the character-based version, they must define each view separately. This technique creates three separate files. The converter does not attempt to decipher each of these files to check if they are for the same query; it simply converts each file into its graphical equivalent. The Results administrator must eliminate “duplicate” files.

Queries

The converter uses the character-based query directory file as an index to find all the user queries when it converts them.

5.1.2 Unconverted Information

This section describes the information that the Converter does not convert when it converts an application. The Converter does not convert the following information to the graphical configuration file:

- Color settings
- Application programmer value-added functions and code
- `results.l` file

Color Settings

The Converter does not convert any color settings. Instead, it assumes that the color information is specified in the `progress.ini` file.

Application Programmer Value-added Functions and Code

Value-added functions allow application programmers to add value-added code. The Converter does not convert any of these functions, nor does it convert the actual value-added code.

results.l File

In character-based Results, this file contains the language and the delimiters for headers and footers. Because the scope of this information is larger in graphical Results, the Converter does not convert this file. If you use the `RESULTS.L` file in your application, you must manually change this file.

5.1.3 Additional Graphical Information

Because the graphical and character-based versions of Results are different, the Converter automatically includes the proper defaults for new information stored in the graphical configuration file.

5.2 Converting Applications

This section describes the conversion process in four sections:

- Starting the Converter
- Converting only the configuration file
- Converting only the query files
- Converting all files

5.2.1 Starting the Converter

When you run the Converter, you must connect to a database. If you do not specify the database connection information when you start the Converter, it will prompt you to do so when you start it.

You can access the Converter in two ways:

- Run the following command in the Procedure Editor:

```
run _resconv.p
```

- Create a Windows icon, then enter the following command in the icon command line:

```
_prowin [ dbspec ] -p _resconv.p [ parameters ]
```

The *dbspec* argument represents a database connection specification with the following syntax:

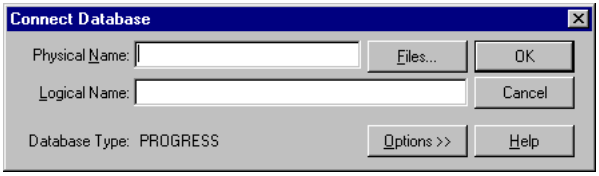
```
-db dbname [ -1 ] [ parameters ]
```

The Physical Database Name (-db) parameter allows you to specify the physical name of a database to connect. All databases specified after the first database in a startup command connect as multi-user databases, regardless of the startup command. The Single-user Mode (-1) parameter designates a single-user database connection. You can also specify parameters that describe how Progress connects to a database (for example, identification of the host and server name of a multi-user database on a network, etc.). The *parameters* argument represents any other startup parameter required to configure the current Progress session to meet your needs.

For more information about the startup parameters referenced in this section, see the *Progress Startup Command and Parameter Reference*.

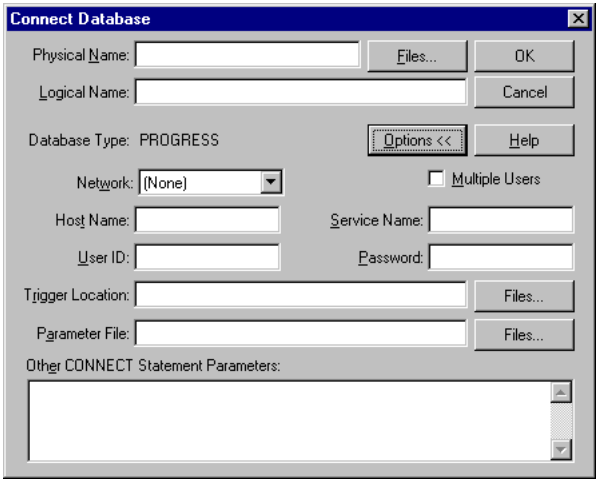
Follow these steps to start the Results Converter and connect to a database:

1 ♦ Start the Converter. If you do not specify the database name on the command line at startup, the Connect Database dialog box appears:

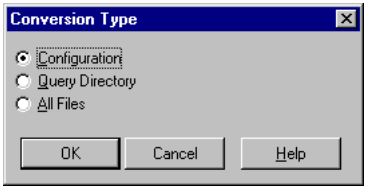


2 ♦ Specify the database name in the Physical Name field.

3 ♦ Choose Options. The Connect Database dialog box expands to provide more options:



4 ♦ Choose OK. The Converter appears and displays the Conversion Type dialog box:



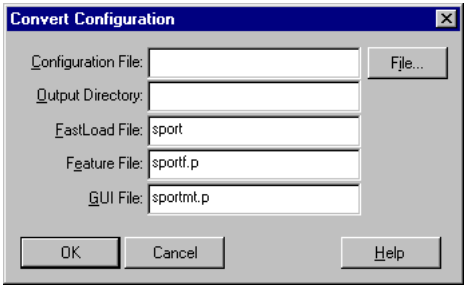
The Conversion Type dialog box provides three choices:

- **Configuration** — Converts the configuration file.
- **Query Directory** — Converts the query directory and its files.
- **All Files** — Converts the configuration file and all the query files.

5.2.2 Converting Only the Configuration File

Follow these steps to convert only the configuration file:

1 ♦ Choose Configuration, then choose OK. The Convert Configuration dialog box appears:



2 ♦ Specify the name of the configuration file you want to convert.

3 ♦ Specify the output directory where you want to output the new file.

If you do not specify an output directory, the Converter places the converted configuration, fastload, feature, and GUI files in the current working directory.

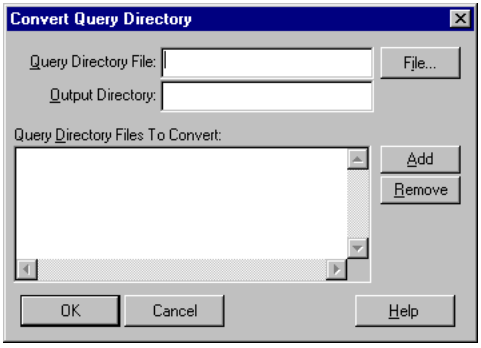
4 ♦ Change the fastload, feature, or GUI filenames if you do not want to use the defaults. The default values are taken from the logical name of the first connected database. Fastload filenames can be up to five characters long; feature and GUI filenames can be up to eleven characters long.

5 ♦ Choose OK. The Converter converts the configuration file to a graphical file. When the file is converted, the Converter main window appears.

5.2.3 Converting Only the Query Files

Follow these steps to convert only the query files:

1 ♦ Choose Query Directory, then choose OK. The Convert Query Directory dialog box appears:



2 ♦ Specify the query directory file you want to convert.

3 ♦ Specify the output directory where you want to output the new file.

4 ♦ If you do not specify an output directory, the Converter places the converted query files and query directories in the same directory as the original, character-based query directory file.

5 ♦ Choose Add. The Converter lists the specified query directory file in the Query Directories To Convert list.

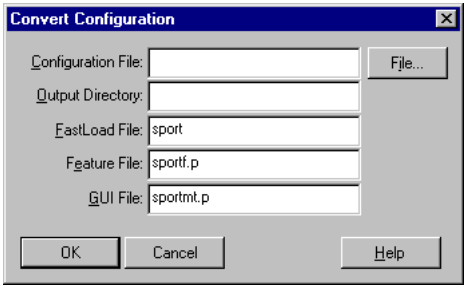
6 ♦ Add additional query directory files in the same manner.

7 ♦ Choose OK. The Converter converts the specified query directories to graphical files. When the files are converted, the Converter main window appears.

5.2.4 Converting Both the Configuration and Query Files

Follow these steps to convert the configuration file and the query files.

1 ♦ Choose All Files, then choose OK. The Convert Configuration dialog box appears:



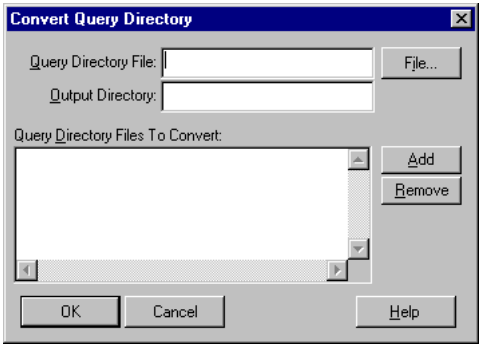
2 ♦ Specify the name of the configuration file you want to convert.

3 ♦ Specify the output directory where you want to output the new file.

If you do not specify an output directory, the Converter places the converted configuration file in the current working directory.

4 ♦ Change the fastload, feature, or GUI filenames if you do not want to use the defaults.

5 ♦ Choose OK. The Converter converts the configuration file to a graphical file. When the configuration file is converted, the Convert Query Directory dialog box appears:



6 ♦ Specify the query directory file you want to convert in the Query Directory field.

7 ♦ Specify the output directory to which you want to output the new file in the Output Directory field.

If you do not specify an output directory, the Converter places the converted query files and query directories in the same directory as the original, character-based Query Directory file.

8 ♦ Choose Add. The Converter lists the specified query directory files in the Query Directories To Convert list box.

9 ♦ Add all the desired query directory files in the same manner.

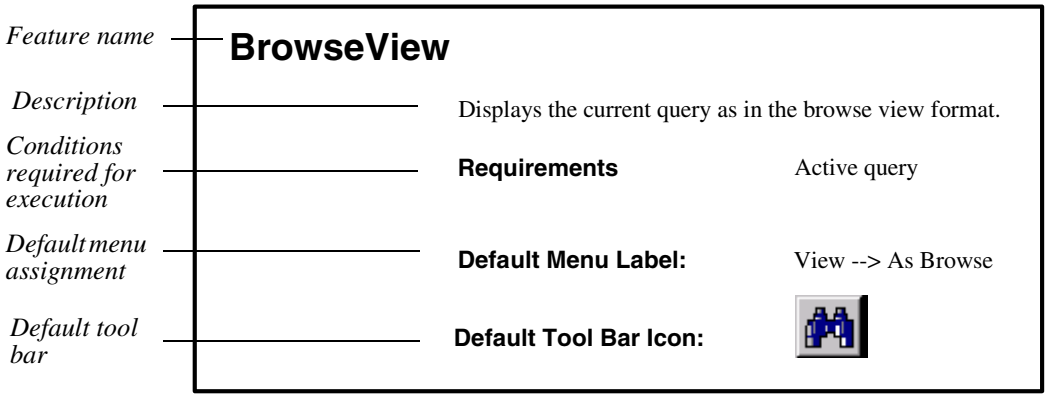
10 ♦ Choose OK. The Converter converts the specified query directories to graphical files. When the files are converted, the Converter main window appears.

A

Results Features

This appendix contains an alphabetical listing of reference entries for each feature in Results. Use this reference to redesign the default menu system and tool bar layout, set feature security, or execute features from a feature procedure using the API `sffire.p` procedure. See Chapter 3, "Programming Results," for more information about API procedures and executing features from feature procedures.

The following example provides a description of the types of information supplied for each reference entry:



A.1 AboutResults

Displays the About Results dialog box. This dialog box displays information about the current product installation.

Requirements: None.

Default Menu Label: Help→ About Results

Default Tool Bar Icon: None. Not on the default tool bar.

A.2 AdminCustomizeInterface

Displays the Preferences dialog box. This dialog box allows you to set the product name displayed in the application window title bar, set the icon image for the application, and set the Results exit behavior.

Requirements: None.

Default Menu Label: Query→ Customize→ Preferences

Default Tool Bar Icon: None. Not on the default tool bar.

A.3 AdminDeployment

Displays the Deployment dialog box. This dialog box allows you to set the name of the fastload file, feature definition file, and graphical configuration file.

Requirements: None.

Default Menu Label: Query→ Customize→ Deployment

Default Tool Bar Icon: None. Not on the default tool bar.

A.4 AdminExportType

Displays the Standard Export dialog box. This dialog box allows you to define the standard export formats available for the current site.

Requirements: None.

Default Menu Label: Query→ Site Admin→ Standard Export Format

Default Tool Bar Icon: None. Not on the default tool bar.

A.5 AdminFeatures

Displays the Feature Editor dialog box. This dialog box allows you to create, edit, and delete features for Results. You cannot edit or delete Results features from this dialog box.

Requirements: None.

Default Menu Label: Query→ Customize→ Features

Default Tool Bar Icon: None. Not on the default tool bar.

A.6 AdminIntegration

Displays the Integration Procedures dialog box. This dialog box allows you to integrate Progress procedures into the Results engine to customize Results data security and interface management processing.

Requirements: None.

Default Menu Label: Query→ Customize→ Integration Procedures

Default Tool Bar Icon: None. Not on the default tool bar.

A.7 AdminLabelField

Displays the Label Field Selection dialog box. This dialog box allows you to define the parameters that determine default label selection for all tables in the connected databases.

Requirements: None.

Default Menu Label: Query→ Site Admin→ Label Field Selection

Default Tool Bar Icon: None. Not on the default tool bar.

A.8 AdminLabelType

Displays the Standard Label dialog box. This dialog box allows you to define the label sizes available for use in the label view.

Requirements: None.

Default Menu Label: Query→ Site Admin→ Standard Label Size

Default Tool Bar Icon: None. Not on the default tool bar.

A.9 AdminMenuEdit

Displays the Menu Editor dialog box. This dialog box allows you to add, edit, and delete menu items from the Results menu structure. You can also use this dialog box to associate features with menu items.

Requirements: None.

Default Menu Label: Query→ Customize→ Menu Layout

Default Tool Bar Icon: None. Not on the default tool bar.

A.10 AdminProgInitialize

Clears the current query and initializes the API shared variables. Use this feature prior to creating a new query in a feature procedure. This feature is designed specifically for use with the API `sffire.p` procedure.

Requirements: None.

Default Menu Label: None. Not in the default menu system.

Default Tool Bar Icon: None. Not on the default tool bar.

A.11 AdminProgInstantiate

Initializes any unset API shared variables and all other API data structures with valid default values. Use this feature prior to executing (displaying, printing, generating, or saving) a query defined or modified in a feature procedure. This feature is designed specifically for use with the API `sffire.p` procedure.

Requirements: None.

Default Menu Label: None. Not in the default menu system.

Default Tool Bar Icon: None. Not on the default tool bar.

A.12 AdminProgWrite4GL

Generates 4GL code for the current query and view, placing the source in a specified ASCII file. This feature is designed specifically for use with the API `sffire.p` procedure.

Requirements: None.

Default Menu Label: None. Not in the default menu system.

Default Tool Bar Icon: None. Not on the default tool bar.

This feature has two arguments:

```
proc-name, code-format
```

The *proc-name* argument specifies the name of an ASCII file where the generated code is stored. The *code-format* argument must be the character g or r. The character g generates stand-alone code. The character r generates code dependent on the Results application. You cannot generate code with the r option if the current view is the browse or form view.

A.13 AdminRebuild

Displays the Application Rebuild dialog box. This dialog box allows you to rebuild the database query configuration file and all query files when you change the schema of an application database.

Requirements: None.

Default Menu Label: Query→ Site Admin→ Application Rebuild

Default Tool Bar Icon: None. Not on the default tool bar.

A.14 AdminReportType

Displays the Standard Page Size dialog box. This dialog box allows you to define the page sizes available for use in the report view.

Requirements: None.

Default Menu Label: Query→ Site Admin→ Standard Page Size

Default Tool Bar Icon: None. Not on the default tool bar.

A.15 AdminReset

Displays the Reset dialog box. This dialog box allows you to reset the Results menu system and tool bar to their default states. It also allows you to delete all user-defined features in the current installation of Results.

Requirements: None.

Default Menu Label: Query→ Customize→ Reset

Default Tool Bar Icon: None. Not on the default tool bar.

A.16 AdminSecurity

Displays the Feature Security dialog box. This dialog box allows you to specify which users can access features defined in Results. For each feature, you can specify a list of user IDs that have access to the feature.

Requirements: None.

Default Menu Label: Query→ Site Admin→ Feature Security

Default Tool Bar Icon: None. Not on the default tool bar.

A.17 AdminSubMenu

Represents the Site Administration submenu and cannot be executed from a feature procedure using the API `sffire.p` procedure. This menu provides access to the following administration features: AdminExportType, AdminLabelField, AdminLabelType, AdminRebuild, AdminReportType, AdminSecurity, AdminTableAlias, AdminTableRelations, and AdminWhere.

Requirements: None.

Default Menu Label: Query→ Site Admin

Default Tool Bar Icon: None. Not on the default tool bar.

A.18 AdminTableAlias

Displays the Table Alias dialog box. This dialog box allows you to define or delete aliases for tables in the connected databases.

Requirements: None.

Default Menu Label: Query→ Site Admin→ Table Aliases

Default Tool Bar Icon: None. Not on the default tool bar.

A.19 AdminTableRelations

Displays the Table Relationships dialog box. This dialog box allows you to define or limit relationships between tables in the connected databases.

Requirements: None.

Default Menu Label: Query→ Site Admin→ Table Relationships

Default Tool Bar Icon: None. Not on the default tool bar.

A.20 AdminToolBarEdit

Displays the Tool Bar Layout dialog box. This dialog box allows you to add and delete icons on the Results tool bar. You can also use this feature to redefine the features associated with icons on the Results tool bar.

Requirements: None.

Default Menu Label: Query→ Customize→ Tool Bar Layout

Default Tool Bar Icon: None. Not on the default tool bar.

A.21 AdminWhere

Displays the Table Data Selection dialog box. This dialog box allows you to define static selection criteria (WHERE clause) for tables in the connected database to limit access to the data stored in those tables. The static selection criteria specified for a table executes every time you access a record in the table.

Requirements: None.

Default Menu Label: Query→ Site Admin→ Table Data Selection

Default Tool Bar Icon: None. Not on the default tool bar.

A.22 Array

Displays the Stack Array dialog box. This dialog box allows you to format a stacked array.

Requirements: Active query.

Default Menu Label: Field→ Add Calculated Field→ Stacked Array

Default Tool Bar Icon: None. Not on the default tool bar.

A.23 BrowseView

Displays the current query, as in the browse view format.

Requirements: Active query.

Default Menu Label: View→ As Browse

Default Tool Bar Icon:



A.24 CalcSubMenu

Represents the Add Calculated Field submenu and cannot be executed from a feature procedure using the API `sffire.p` procedure. This menu provides access to the following features: PercentTotal, RunningTotal, Counter, Array, Lookup, Math, StringFunc, NumericFunc, DateFunc, and LogicalFunc.

Requirements: Active query.

Default Menu Label: Field→ Add Calculated Field

Default Tool Bar Icon: None. Not on the default tool bar.

A.25 Counter

Displays the Counter dialog box. This dialog box allows you to define a counter field for the current query.

Requirements: Active query and any view format except the label view.

Default Menu Label: Field→ Add Calculated Field→ Counter

Default Tool Bar Icon: None. Not on the default tool bar.

A.26 CustomLabel

Displays the Custom Label dialog box. This dialog box allows you to define the dimensions of the label view for the active query.

Requirements: Active query and the label view.

Default Menu Label: Options→ Custom Label

Default Tool Bar Icon: None. Not on the default tool bar.

A.27 CustomReport

Displays the Custom Page Size dialog box. This dialog box allows you to define line spacing, column spacing, page length, left margin, and lines between the report sections for the report view of the active query.

Requirements: Active query and the report view.

Default Menu Label: Options→ Custom Page Size

Default Tool Bar Icon: None. Not on the default tool bar.

A.28 DataGovernor

Displays the Data Governor dialog box. This dialog box allows you to limit the number of records a query displays.

Requirements: Active query.

Default Menu Label: Data→ Governor

Default Tool Bar Icon: None. Not on the default tool bar.

A.29 DateFunc

Displays the Date Function dialog box. This dialog box allows you to define an expression that yields a date value represented by a field in the active query.

Requirements: Active query.

Default Menu Label: Field→ Add Calculated Field→ Date Function

Default Tool Bar Icon: None. Not on the default tool bar.

A.30 Exit

Exits Results. If you modified the active query and did not save it, this feature displays an alert box notifying you of the changes and prompting you to save your changes to the query. Exit behavior is governed by the AdminCustomizeInterface feature.

Requirements: None.

Default Menu Label: Query→ Exit

Default Tool Bar Icon: None. Not on the default tool bar.

A.31 ExportSubMenu

This feature represents the Custom Export submenu and cannot be executed from a feature procedure using the API `sffire.p` procedure. This menu provides access to the following custom export features: FieldDelimiters, FieldSeparators, FixedWidthFields, OutputHeaderRecord, RecordEnd, and RecordStart.

Requirements: Active query and the export view.

Default Menu Label: Options→ Custom Export

Default Tool Bar Icon: None. Not on the default tool bar.

A.32 ExportView

Displays the current query, as in the export view format.

Requirements: Active query.

Default Menu Label: View→ As Export

Default Tool Bar Icon:



A.33 FieldDelimiters

Displays the Field Delimiters dialog box. This dialog box allows you to specify the delimiter characters for fields in a data export. Delimiter characters begin and end each field in the export.

Requirements: Active query and the export view.

Default Menu Label: Options→ Custom Export→ Field Delimiters

Default Tool Bar Icon: None. Not on the default tool bar.

A.34 FieldProperties

Displays the Properties dialog box. This dialog box allows you to manipulate the display format, label, any underlying expression of a field in the active query.

Requirements: Active query and any view format except the label view.

Default Menu Label: Field→ Properties

Default Tool Bar Icon: None. Not on the default tool bar.

A.35 Fields

Displays the Add/Remove Fields dialog box for the active query in the browse, form, report, or export view. The Add/Remove Fields dialog box allows you to redefine the fields included in the active query. The Fields feature displays the Label Layout dialog box for the active query in the label view. This feature requires an active query in Results.

Requirements: Active query.

Default Menu Label: Field→ Add/Remove Fields

Default Tool Bar Icon: None. Not on the default tool bar.

A.36 FieldSeparators

Displays the Field Separators dialog box. This dialog box allows you to specify the character used to separate the fields of a data export.

Requirements: Active query and the export view.

Default Menu Label: Options→ Custom Export→ Field Separators

Default Tool Bar Icon: None. Not on the default tool bar.

A.37 FileClose

Closes the active query. If the active query is new or has some unsaved modifications, Results uses an alert box to prompt you to save the active query.

Requirements: Active query.

Default Menu Label: Query→ Close

Default Tool Bar Icon: None. Not on the default tool bar.

A.38 FileDelete

Displays the Delete dialog box. This dialog box displays a list of queries defined in the current query directory. You can select one or more queries to delete.

Requirements: None.

Default Menu Label: Query→ Delete

Default Tool Bar Icon: None. Not on the default tool bar.

A.39 FileOpen

Displays the Open dialog box. This dialog box allows you to select a query to open from the current query directory, a public query directory, or a query directory of another user. The FileOpen feature closes an existing active query prior to opening a new query.

Requirements: None.

Default Menu Label: Query→ Open

Default Tool Bar Icon:



A.40 FileSave

Saves the active query in the current query directory. If the active query is new, Results displays the Save As dialog box to allow you to name and save the active query.

Requirements: Active query.

Default Menu Label: Query→ Save

Default Tool Bar Icon:



A.41 FileSaveAs

Displays the Save As dialog box. This dialog box allows you to save the active query under a specified name in the current query directory.

Requirements: Active query.

Default Menu Label: Query→ Save As

Default Tool Bar Icon: None. Not on the default tool bar.

A.42 FixedWidthFields

Toggles between fixed and variable field lengths for data export operations. In a variable-length data export, the length of the data within a column varies across the rows in the export. The maximum width of the data in the column is governed by the Format phrase specified for the column in the Data Dictionary. In a fixed-width data export, the character length within columns is constant across the rows in the export. The maximum width of the data in the column is governed by the Format phrase specified for the column in the Data Dictionary. Values that do not fill the maximum width are padded with trailing space characters.

Requirements: Active query and the export view.

Default Menu Label: Options→ Custom Export→ Fixed-width Fields

Default Tool Bar Icon: None. Not on the default tool bar.

A.43 FormView

Displays the current query as in the form view format.

Requirements: Active query.

Default Menu Label: View→ As Form

Default Tool Bar Icon:



A.44 FrameProperties

Displays the Frame Properties dialog box. This dialog box allows you to specify a title and the vertical position of a frame associated with the browse view or form view of an active query. You can also use the Frame Properties dialog box to arrange multiple frames on the display and to designate a frame as read only.

Requirements: Active query and the form view or the browse view.

Default Menu Label: Options→ Frame Properties

Default Tool Bar Icon: None. Not on the default tool bar.

A.45 Generate

Displays the Generate 4GL dialog box. This dialog box allows you to specify the name of an ASCII file to contain the stand-alone Progress source code generated for the active query and view in the Results product.

Requirements: Active query.

Default Menu Label: Query→ Generate

Default Tool Bar Icon: None. Not on the default tool bar.

A.46 HeadersAndFooters

Displays the Headers and Footers dialog box. This dialog box allows you to position, format, and edit the content of report headers and footers.

Requirements: Active query and the report view.

Default Menu Label: Options→ Headers and Footers

Default Tool Bar Icon: None. Not on the default tool bar.

A.47 HelpContents

Displays the main table of contents topic of the Results help file in the help viewer.

Requirements: None.

Default Menu Label: Help→ Contents

Default Tool Bar Icon: None. Not on the default tool bar.

A.48 HelpSearch

Displays the Search dialog box of the help viewer. This dialog box allows you to search for help information using keywords.

Requirements: None.

Default Menu Label: Help→ Search for Help On

Default Tool Bar Icon: None. Not on the default tool bar.

A.49 HelpUse

Displays information in the help viewer about how to use the Progress help system.

Requirements: None.

Default Menu Label: Help→ How to Use Help

Default Tool Bar Icon: None. Not on the default tool bar.

A.50 Includes

Displays the Available Table dialog box followed by the Insert Code dialog box. The Available Table dialog box allows you to select a table referenced in the active query. Once you select a table, the Insert Code dialog box allows you to specify Progress code to insert after the FOR statement associated with the selected table in the active query.

Requirements: Active query.

Default Menu Label: None. Not in the default menu system.

Default Tool Bar Icon: None. Not on the default tool bar.

A.51 Information

Displays the Query Information dialog box. This dialog box displays information about the active query, including the tables and fields referenced in the query.

Requirements: Active query.

Default Menu Label: Options→ Query Information

Default Tool Bar Icon: None. Not on the default tool bar.

A.52 Joins

Displays the Relationship Types dialog box to define the characteristics of a join in the active query.

Requirements: Active query with at least two tables.

Default Menu Label: Table→ Relationship Types

Default Tool Bar Icon: None. Not on the default tool bar.

A.53 LabelView

Displays the label view associated with the active query.

Requirements: Active query.

Default Menu Label: View→ As Label

Default Tool Bar Icon:



A.54 LogicalFunc

Displays the Logical Function dialog box. This dialog box allows you to define an expression that yields a logical value represented by a field in the active query.

Requirements: Active query.

Default Menu Label: Field→ Add Calculated Field→ Logical Function

Default Tool Bar Icon: None. Not on the default tool bar.

A.55 Lookup

Displays the Lookup Source Field dialog box. This dialog box allows you to define a lookup field to display in the active query. A lookup field is a field in the active query that you can use to establish a one-to-one relationship with a new table not in the active query to display one or more fields from the new table as part of the active query.

Requirements: Active query.

Default Menu Label: Field→ Add Calculated Field→ Lookup Field

Default Tool Bar Icon: None. Not on the default tool bar.

A.56 MasterDetail

Displays the Master-Detail dialog box. This dialog box allows you to set a visual break point between tables on a report, browse, or form view.

Requirements: Active query with at least two tables and the browse, form, or report view.

Default Menu Label: Options→ Master-Detail

Default Tool Bar Icon: None. Not on the default tool bar.

A.57 Math

Displays the Math dialog box. This dialog box allows you to define an expression that yields a numeric value represented by a field in the active query.

Requirements: Active query.

Default Menu Label: Field→ Add Calculated Field→ Math

Default Tool Bar Icon: None. Not on the default tool bar.

A.58 Messages

Displays the Messages dialog box. This dialog box allows you to get information about a specified Progress error message.

Requirements: None.

Default Menu Label: Help→ Messages

Default Tool Bar Icon: None. Not on the default tool bar.

A.59 NewBrowseView

Displays the Add/Remove Tables dialog box to allow you to define a new query and display it in the browse view.

Requirements: None.

Default Menu Label: Query→ New→ Browse

Default Tool Bar Icon: None. Not on the default tool bar.

A.60 NewDuplicateView

Displays the Add/Remove Tables dialog box to allow you to define a new query. The query appears on the display in the view mode of the last active query.

Requirements: Active query.

Default Menu Label: None. Not in the default menu system.

Default Tool Bar Icon:



A.61 NewExportView

Displays the Add/Remove Tables dialog box to allow you to define a new query and display it in the export view.

Requirements: None.

Default Menu Label: Query→ New→ Export

Default Tool Bar Icon: None. Not on the default tool bar.

A.62 NewFormView

Displays the Add/Remove Tables dialog box to allow you to define a new query and display it in the form view.

Requirements: None.

Default Menu Label: Query→ New→ Form

Default Tool Bar Icon: None. Not on the default tool bar.

A.63 NewLabelView

Displays the Add/Remove Tables dialog box to allow you to define a new query and display it in the label view.

Requirements: None.

Default Menu Label: Query→ New→ Label

Default Tool Bar Icon: None. Not on the default tool bar.

A.64 NewReportView

Displays the Add/Remove Tables dialog box to allow you to define a new query and display it in the report view.

Requirements: None.

Default Menu Label: Query→ New→ Report

Default Tool Bar Icon: None. Not on the default tool bar.

A.65 NewSubMenu

Provides access to the following features: NewBrowseView, NewExportView, NewFormView, NewLabelView, and NewReportView. This feature represents the New submenu and cannot be executed from a feature procedure using the API `sffire.p` procedure.

Requirements: None.

Default Menu Label: Query→ New

Default Tool Bar Icon: None. Not on the default tool bar.

A.66 NumericFunc

Displays the Numeric Function dialog box. This dialog box allows you to define an expression that yields a numeric value represented by a field in the active query.

Requirements: Active query.

Default Menu Label: Field→ Add Calculated Field→ Numeric Function

Default Tool Bar Icon: None. Not on the default tool bar.

A.67 OutputHeaderRecord

Toggles Output Headers on or off for data export operations.

Requirements: Active query and the export view.

Default Menu Label: Options→ Custom Export→ Output Header Record

Default Tool Bar Icon: None. Not on the default tool bar.

A.68 PageBreak

Displays the Page Break dialog box. This dialog box allows you to select a previously defined sort field in the active query to govern page breaks.

Requirements: Active query with sorted fields and the report view.

Default Menu Label: Options→ Page Break

Default Tool Bar Icon: None. Not on the default tool bar.

A.69 PercentTotal

Displays the Percent of Total dialog box. This dialog box allows you to select a numeric field from a table defined in the active query to be used in a percentage of total calculation.

Requirements: Active query.

Default Menu Label: Field→ Add Calculated Field→ Percent of Total

Default Tool Bar Icon: None. Not on the default tool bar.

A.70 PrintClip

Executes the active query and sends the output to the system clipboard.

Requirements: Active query.

Default Menu Label: Query→ Print→ To Clipboard

Default Tool Bar Icon: None. Not on the default tool bar.

A.71 PrintFile

Displays the Output to File dialog box. This dialog box allows you to specify the name of an ASCII file. Once you specify an output file, the active query executes and the output goes to that file.

Requirements: Active query.

Default Menu Label: Query→ Print→ To File

Default Tool Bar Icon: None. Not on the default tool bar.

A.72 PrintPreview

Executes the active query and displays the output on screen in the Print Preview dialog box.

Requirements: Active query.

Default Menu Label: Query→ Print Preview

Default Tool Bar Icon:



A.73 PrintPrinter

Executes the active query and prints the query output. In Windows, this feature displays the Print dialog box to allow the user to specify a different printer.

Requirements: Active query.

Default Menu Label: Query→ Print→ To Printer

Default Tool Bar Icon: None. Not on the default tool bar.

A.74 PrintPrinterNoBox

Executes the active query and prints the query output to the default system printer. In Windows, this feature sends the output to the default system printer.

Requirements: Active query.

Default Menu Label: None. Not in the default menu system.

Default Tool Bar Icon:



A.75 PrintSubMenu

Provides access to the following features: PrintClip, PrintFile, and PrintPrinter. This feature represents the Print submenu and cannot be executed from a feature procedure using the API `sffire.p` procedure.

Requirements: Active query.

Default Menu Label: Query→ Print

Default Tool Bar Icon: None. Not on the default tool bar.

A.76 ProgrammingSubMenu

Provides access to the following features: AdminCustomizeInterface, AdminDeployment, AdminFeatures, AdminIntegration, AdminMenuEdit, AdminReset, and AdminToolBarEdit. This feature represents the Customize submenu and cannot be executed from a feature procedure using the API `sffire.p` procedure.

Requirements: Active query.

Default Menu Label: Query→ Customize

Default Tool Bar Icon: None. Not on the default tool bar.

A.77 ReadOtherDirectory

No functionality associated with this feature. The primary use for this feature is to provide control over user access to query directories through the Feature Security mechanism. It is not designed for use with the API `sffire.p` procedure.

Requirements: None.

Default Menu Label: None. Not in the default menu system.

Default Tool Bar Icon: None. Not on the default tool bar.

A.78 ReadPublicDirectory

No functionality associated with this feature. The primary use for this feature is to provide control over user access to query directories through the Feature Security mechanism. It is not designed for use with the API `sffire.p` procedure.

Requirements: None.

Default Menu Label: None. Not in the default menu system.

Default Tool Bar Icon: None. Not on the default tool bar.

A.79 ReAsk

Displays the Ask At Run Time dialog box to prompt for a value to use in the selection criteria associated with the current query.

Requirements: Active query with defined selection criteria that prompts for a value at run time (Ask At Run Time). The current view must be the browse or form view.

Default Menu Label: Data→ Re-ask Questions

Default Tool Bar Icon: None. Not on the default tool bar.

A.80 RecentMessages

Displays the Recent Messages dialog box. This dialog box allows you to get information about recent Progress error messages.

Requirements: None.

Default Menu Label: Help→ Recent→ Messages

Default Tool Bar Icon: None. Not on the default tool bar.

A.81 RecordAdd

No functionality associated with this feature. Use this feature to allow or deny a user, through the Feature Security mechanism, permission to add database records using a Results query in the form view. It is not designed for use with the API `sffire.p` procedure.

Requirements: None.

Default Menu Label: None. Not in the default menu system.

Default Tool Bar Icon: None. Not on the default tool bar.

A.82 RecordDelete

No functionality associated with this feature. Use this feature to allow or deny a user, through the Feature Security mechanism, permission to delete database records using a Results query in the form view. It is not designed for use with the API `sffire.p` procedure.

Requirements: None.

Default Menu Label: None. Not in the default menu system.

Default Tool Bar Icon: None. Not on the default tool bar.

A.83 RecordEnd

Displays the Record End dialog box. This dialog box allows you to specify the character used to signal the end of a record in the output of a data export operation.

Requirements: Active query and the export view.

Default Menu Label: Options→ Custom Export→ Record End

Default Tool Bar Icon: None. Not on the default tool bar.

A.84 RecordStart

Displays the Record Start dialog box. This dialog box allows you to specify the character used to signal the start of a record in the output of a data export operation.

Requirements: Active query and the export view.

Default Menu Label: Options→ Custom Export→ Record Start

Default Tool Bar Icon: None. Not on the default tool bar.

A.85 RecordUpdate

No functionality associated with this feature. Use this feature to allow or deny a user, through the Feature Security mechanism, permission to update database records using a Results query in the form view. It is not designed for use with the API `sffire.p` procedure.

Requirements: None.

Default Menu Label: None. Not in the default menu system.

Default Tool Bar Icon: None. Not on the default tool bar.

A.86 ReportView

Displays the report view associated with the active query.

Requirements: Active query.

Default Menu Label: View→ As Report

Default Tool Bar Icon:



A.87 RunningTotal

Displays the Running Totals dialog box. This dialog box allows you to select a database field from a table in the active query and use it to define a calculated field that will keep a running total for the selected database field across the records returned in the active query.

Requirements: Active query.

Default Menu Label: Field→ Add Calculated Field→ Running Total

Default Tool Bar Icon: None. Not on the default tool bar.

A.88 Selection

Displays the Available Tables dialog box. From this dialog box, you can select a table in the active query and construct a WHERE clause that refines the data selection criteria associated with the selected table.

Requirements: Active query.

Default Menu Label: Data→ Selection

Default Tool Bar Icon:



A.89 SortOrdering

Displays the Sort Order Fields dialog box. This dialog box allows you to select the fields to use when sorting the data output of the active query.

Requirements: Active query.

Default Menu Label: Data→ Sort Ordering

Default Tool Bar Icon:



A.90 StandardExport

Displays the Standard Export dialog box containing a list of predefined export formats. This dialog box allows you to select and assign a predefined export format for the export view of the active query.

Requirements: Active query and the export view.

Default Menu Label: Options→ Standard Export

Default Tool Bar Icon: None. Not on the default tool bar.

A.91 StandardLabel

Displays the Standard Label dialog box containing a list of predefined label formats. This dialog box allows you to select and assign a predefined label format for the label view of the active query.

Requirements: Active query and the label view.

Default Menu Label: Options→ Standard Label

Default Tool Bar Icon: None. Not on the default tool bar.

A.92 StandardReport

Displays the Standard Page Size dialog box containing a list of predefined page sizes. This dialog box allows you to select and assign a predefined page size for the report view of the active query.

Requirements: Active query and report view.

Default Menu Label: Options→ Standard Page Size

Default Tool Bar Icon: None. Not on the default tool bar.

A.93 StatusLine

Toggles the display of the status line located at the bottom of the Results application window.

Requirements: Active query.

Default Menu Label: View→ Status Line

Default Tool Bar Icon: None. Not on the default tool bar.

A.94 StringFunc

Displays the String Function dialog box. This dialog box allows you to define an expression that yields a character value represented by a field in the active query.

Requirements: Active query.

Default Menu Label: Field→ Add Calculated Field→ String Function

Default Tool Bar Icon: None. Not on the default tool bar.

A.95 Tables

Displays the Add/Remove Tables dialog box. This dialog box allows you to insert or remove tables from the active query.

Requirements: Active query.

Default Menu Label: Table→ Add/Remove Tables

Default Tool Bar Icon: None. Not on the default tool bar.

A.96 ToolBar

Toggles the display of the tool bar located underneath the menu bar of the Results application window.

Requirements: Active query.

Default Menu Label: View→ Tool Bar

Default Tool Bar Icon: None. Not on the default tool bar.

A.97 Totals

Displays the Field Aggregates dialog box. This dialog box allows you to select a field defined for the active query and define a totals operation for the selected field. You can specify the total for the field to display after a break group or in the summary section of the report output.

Requirements: Active query and the report view.

Default Menu Label: Field→ Aggregates

Default Tool Bar Icon: None. Not on the default tool bar.

A.98 TotalsOnly

Displays the Totals Only Summary dialog box. This dialog box allows you to define report sections as summarized. You can select one or more fields to summarize based on the last field chosen in the sort list of the active query.

Requirements: Active query with sorted fields and the report view.

Default Menu Label: Options→ Totals Only Summary

Default Tool Bar Icon: None. Not on the default tool bar.

A.99 WriteOtherDirectory

No functionality associated with this feature. The primary use for this feature is to provide control over user access to query directories through the Feature Security mechanism. It is not designed for use with the API `sffire.p` procedure.

Requirements: None.

Default Menu Label: None. Not in the default menu system.

Default Tool Bar Icon: None. Not on the default tool bar.

A.100 WritePublicDirectory

No functionality associated with this feature. The primary use for this feature is to provide control over user access to query directories through the Feature Security mechanism. It is not designed for use with the API `sffire.p` procedure.

Requirements: None.

Default Menu Label: None. Not in the default menu system.

Default Tool Bar Icon: None. Not on the default tool bar.

B

Results Limits

This appendix lists the limits that exist within Results. For information on other Progress limits, see the respective document:

- **Database limits; length limits for names** —*Progress Database Administration Guide and Reference*
- **Data value limits** — *Progress Language Reference*
- **Procedure Editor limits** — *Progress Basic Development Tools* (Character only; information for Windows is in online help)
- **Client input/output and sorting limits; name-length limits; compiler limits** — *Progress Client Deployment Guide*

Table B–1 lists the limitations for Results views.

Table B–1: View Limitations

–	Limits
Lines in a mailing label	1-66
Fields in a browse, form, report, or export view	1-64
Header and footer lines	1-8 per position

Table B–2 lists the limitations for Results queries.

Table B–2: Query Limitations

–	Limits
Characters in a query name	1-48
Fields in a query	1-64
Query definitions in a single query directory (qd7) file	1-256
Query source files (`que#####.p`) in a single operating system directory	1-99,999
QD7 files in a single operating system directory	1-390
Sort fields per table	1-16
Characters in the selection criteria (WHERE clause) of a table	Based on Stash Area (-stsh) parameter value (1-31, default 2) or Stack Size (-s) parameter value (3-63, default 63)
Stored custom table relationships (WHERE relationships)	Available disk space
Stored natural table relationships	Available disk space
Table WHERE clauses	Available disk space

Table B–3 lists miscellaneous Results limitations.

Table B–3: Miscellaneous Limitations

–	Limits
Print Preview pages	1- 1,024
Printers	Feature memory space
Databases	Up to 32K (size of character fields)
Fields in field picker	1-512 per table
Tool bar icons	1-14

C

Results Files

This appendix provides the following information about important files associated with the Results product:

- Files generated by Results
- File formats and editing tips

C.1 Files Generated by Results

Table C–1 describes the files generated by Results.

Table C–1: Files Generated by Results *(1 of 2)*

File	Description
database.qc7	A query configuration file (QC7). The *query configuration* file contains the default file relationships, security settings, label settings, data export settings, and report settings for a Results application. The *database* value is the name of the first application database connected for the Results application. Results cannot run without the QC7 file. For more information about query configuration files, see Chapter 2, "Administering Results."This appendix provides information about editing a QC7 file.
database.r	The main fastload procedure for a Results application. The fastload system optimizes the loading of information from the current QC7 file into Results at startup. The *database* value is the base string specified in the Deployment dialog box for the generation of fastload files for a Results application. The default base string is the first five characters of the database name used to name the current QC7 file. For more information about the fastload system for a Results application, see Chapter 3, "Programming Results."
base###.r *baser*.r	Secondary fastload files. Base is the base string specified in the Deployment dialog box for the generation of fastload files for a Results application. The default base string is the first five characters of the database name used to name the current QC7 file.
basef.p *basef*.r	The feature files of a Results application. The feature file is a Progress procedure file in the fastload system that defines and loads features into Results at startup. The *basef*.r file is **not** a compiled version of the *basef*.p file. It is optimized for fast loads and accesses internal data structures of the Results product. The source code in the *basef*.p file performs that same function as the *basef*.r file, only slower. It is supplied to support the deployment process of a Results application. For more information about the fastload system for a Results application, see Chapter 3, "Programming Results."

Table C–1: Files Generated by Results *(2 of 2)*

File	Description
*base*mt.p *base*mt.r	The user interface (UI) files of a Results application. The UI file is a Progress procedure file in the fastload system that configures the menu system and tool bar of Results at startup. The *base*mt.r is not a compiled version of the *base*mt.p file. It is optimized for fast loads and accesses internal data structures of the Results product. The source code in the *base*mt.p file performs that same function as the *base*mt.r file, only slower. It is supplied to support the deployment process of a Results application. For more information about the fastload system for a Results application, see Chapter 3, "Programming Results."
public.qd7 *userid*.qd7 results.qd7	Query directory files. The public query directory (public.qd7) contains query definitions accessible by all or a select group of users. Results also automatically generates a query directory to store the query definitions of a user (*userid*.qd7). It uses the user's user ID to name the query directory file. For users who have the blank user ID, Results creates the results.qd7 query directory file to store the query definitions of that user.
que#####.p	Source files for query definitions in the query definitions files located in the current directory. Results creates query source files in the directory that contains the corresponding query definition. Query source files are five-digit numbers (*#####*). Results uses these numbers to maintain the queries. If you delete a query source file, you cannot open the corresponding query in Results.
database.ql	A query log file. Results uses this file to record status and error messages that occur when you build the query configuration file or open a query. For more information about the query log file, see Chapter 2, "Administering Results."
base.txt	The deployment report for a Results application. This file lists the files to deploy for a Results application. *Base* is the base string specified in the Deployment dialog box for the generation of fastload files for a Results application. For more information about the deployment report, see Chapter 4, "Deploying Results Applications."
_qbf####.d _qbfq####.p _qbf####.p temp.p	Temporary files that Results uses to store information. As you exit Results, it automatically deletes these files. But if your system crashes, they could remain on your disk. You can delete them manually.

C.2 Editing the results.l File

The `results.l` file resides in the `aderes` directory in the Progress product directory (`$DLC\GUI`). It contains the following default site settings:

- Characters used to delimit fields in address labels and functions in report headers and footers.
- Standard export format definitions used in the export view of a query.
- Standard label format definitions used in the label view of a query.
- Standard page size definitions used in the report view of a query.

Results uses the `results.l` file to perform the initial build of a QC7 file for a user site. The default settings in the `results.l` file are transferred directly to the QC7 file during the build. For more information about the `results.l` and QC7 files, see Chapter 2, "Administering Results."

The following sections provide more information about the default settings in the `results.l` file. You can use this information to change the delimiter characters and implement new export, label, and page formats.

NOTE: If you introduce errors into the `results.l` file, you might encounter problems during the initial build of a QC7 file. Avoid editing existing export, label, and page formats.

C.2.1 Delimiter Characters

By default, Results uses curly braces { } to delimit functions in report headers and footers. It also uses braces to delimit the fields in address (and other) labels. You can change these default settings by editing the `results.l` file before you perform the initial build. For example, seven-bit character sets use the curly braces for special characters, so you can changes the `results.l` file to specify angle brackets < > as delimiter characters.

To change these delimiter characters, edit the following lines in the `results.l` file before the initial build:

```
left-delim= "{"
right-delim= "}"
```

Do not specify the same character for the left and right delimiter. You can use square brackets [ ] as long as there are no arrays within your reports or labels. You cannot use parentheses ( ) because they interfere with the label generation code.

C.2.2 Standard Export Formats

The `results.l` file contains several lines that define the standard export formats available for use in the export view of a query. Each line that defines an export format has the following syntax.

SYNTAX

```
export[integer]= "code [ | code ] . . . "
```

The *integer* argument is a unique integer value that identifies the export format definition. The integers used to identify export formats must be sequential. The *code* parameter followed by a value defines the export format. All codes specified for an export format must be enclosed within a single set of double quotes and must be separated using a vertical line (|). Table C–2 provides information about the different definitional codes for an export format.

Table C–2: Definitional Codes for Export Formats *(1 of 2)*

Code	Description
1=	Specifies an integer representing the ASCII value of a character to start each line of the export. If you want to specify more than one character, specify a comma-separated list of integers. There is no default character for this setting.
2=	Specifies an integer representing the ASCII value of a character to end each line of the export. If you want to specify more than one character, specify a comma-separated list of integers. There is no default character for this setting.
3=	Specifies an integer representing the ASCII value of a character to delimit each field of the export. If you want to specify more than one character, specify a comma-separated list of integers. There is no default character for this setting.
4=	Specifies an integer representing the ASCII value of a character to separate each field in a line of the export. If you want to specify more than one character, specify a comma-separated list of integers. There is no default character for this setting.
b=	Specifies the base date for a DIF export. The default value is the unknown value (?). If you do not specify a base date, then all dates are exported as character values.

Table C–2: Definitional Codes for Export Formats *(2 of 2)*

Code	Description
d=	Specifies the data types to delimit as fields using the characters specified with the "3=" code. Data type specifications are represented as integer values. 1 CHARACTER 2 DATE 3 LOGICAL 4 INTEGER 5 DECIMAL 6 RAW 7 RECID Do not separate data type integers with commas or spaces. For example, d=123 specifies the CHARACTER, DATE, and LOGICAL data types. The default value is an asterisk (*), representing all data types.
f=	Specifies "y" for a fixed width export. The default value is "n" for a variable-width export.
h=	Specifies "y" to include field headers in the export. The default value is "n."
i=	Specifies "y" if the export program (p=) requires a prepass to count records. The default value is "n."
l=	Specifies a label for the export format. This label represents the format in Results.
p=	Specifies the name of the Progress procedure that generates the export. Results currently supports the following export procedures: `e-pro` Progress export procedure `e-ascii` ASCII export procedure `e-dif` DIF export procedure `e-sylk` SYLK export procedure The `u-export.p` procedure is a sample export procedure. See Chapter 3, "Programming Results," for more information about this procedure.
t=	Specifies a name for the export format. Results uses this name to manage the format, so the name must be unique.

The following excerpt from the `result.l` file shows some export format definitions:

```
export[1]= "t=PROGRESS|p=e-pro|2=32,13,10|3=34|4=32|l=PROGRESS Export"
export[2]= "t=ASCII|p=e-ascii|2=13,10|3=34|4=44|l=Generic ASCII"
export[3]= "t=ASCII-H|p=e-ascii|h=y|2=13,10|3=34|4=44|l=ASCII w/headers"



export[11]= "t=DIF|p=e-dif|l=DIF (w/dates as strings)"



export[15]= "t=SYLK|p=e-sylk|l=SYLK"
```

C.2.3 Standard Label Formats

The `results.l` file contains several lines that define the standard label formats available for use in the label view of a query. Each line that defines a label format has the following syntax:

SYNTAX

```
label[integer]= "code [|code] . . ."
```

The *integer* argument is a unique integer value that identifies the label format definition. The integers used to identify label formats must be sequential. The *code* parameter is a special code followed by a value that defines the label format. All codes specified for a label format must be enclosed within a single set of double quotes and must be separated using a vertical line (|). Table C–3 provides information about the different definitional codes for a label format.

Table C–3: Definitional Codes for Label Formats *(1 of 2)*

Code	Description
a=	Specifies the number of labels across the printed page. The default value is 1.
c=	Specifies a name for the label format. This name represents the format in Results. Must be a unique name.
d=	Specifies the width and height of the label in inches. For example, 3-1/2 x 1. Separate the width and height values using an x. The width value must be first. Specifies fractional values by separating the numerator and denominator values with a forward slash (/). If the fractional value is preceded by a whole number, separate the whole number from the fraction using a hyphen (-).

Table C–3: Definitional Codes for Label Formats *(2 of 2)*

Code	Description
h=	Specifies the height of the label in lines.
l=	Specifies the number of lines between labels. The default value is 1.
s=	Specifies the horizontal space between labels in characters. The default value is 0.
w=	Specifies the width of the label in characters.
x=	Specifies the left margin of the paper in characters. The default value is 0.

The following excerpt from the `result.l` file shows some label format definitions:

```
label[1]= "*Standard Label Sizes"
label[2]= "d=3-1/2 x 15/16 in|w=35|h=5|c=1-wide"
label[3]= "d=3-1/2 x 15/16 in|w=35|h=5|a=2|s=2|c=2-wide"
                  .
                  .
                  .
label[14]= "*Avery Catalog Labels"
label[15]= "d=3-1/2 x 15/16 in|w=35|h=5|c=Avery 4145 Address"
label[16]= "d=3-1/2 x 15/16 in|w=35|h=5|c=Avery 4010 Address (roll)"
```

If the first code parameter of a label format definition begins with an asterisk (*), Results ignores the definition and moves on to the next definition.

C.2.4 Standard Page Formats

The `results.l` file contains several lines that define the standard page formats available for use in the report view of a query. Each line that defines a page format has the following syntax:

SYNTAX

```
page[integer]= "code [|code] . . ."
```

The *integer* argument is a unique integer value that identifies the page format definition. The integers used to identify page formats must be sequential. The *code* parameter is a special code followed by a value that defines the page format. All codes specified for a page format must be enclosed within a single set of double quotes and must be separated using a vertical line (|). Table C–4 provides information about the different definitional codes for a page format.

Table C–4: Definitional Codes for Page Formats

Code	Description
c=	Specifies a label for the page format. This label represents the format in Results. Must be a unique name.
d=	Specifies the width and height of the label in inches (in) or millimeters (mm). For example, 210 x 297 mm or 3-1/2 x 1 in. Separate the width and height values using an x. The width value must be first. Specifies fractional values by separating the numerator and denominator with a forward slash (/). If the fractional value is preceded by a whole number, separate the whole number from the fraction using a hyphen (-).
h=	Specifies the page height in lines.
w=	Specifies the page width in characters.

The following excerpt from the `result.l` file shows some page format definitions:

```
page[1]= "d=8-1/2 x 11 in|w=85|h=66|c=Letter"
page[2]= "d=8-1/2 x 14 in|w=85|h=84|c=Legal"
page[3]= "d=11 x 17 in|w=110|h=102|c=Tabloid"
page[4]= "d=9-1/2 x 4-1/8 in|w=95|h=24|c=Envelope-#10"
page[5]= "d=229 x 162 mm|w=90|h=38|c=Envelope-C5"
page[6]= "d=220 x 110 mm|w=86|h=26|c=Envelope-DL"
page[7]= "d=7-1/2 x 3-7/8 in|w=75|h=23|c=Envelope-Monarch"
page[8]= "d=7-1/4 x 10-1/2 in|w=72|h=63|c=Executive"
page[9]= "d=297 x 420 mm|w=117|h=99|c=A3"
page[10]= "d=210 x 297 mm|w=82|h=70|c=A4"
page[11]= "d=148 x 210 mm|w=58|h=49|c=A5"
page[12]= "d=182 x 257 mm|w=71|h=60|c=B5"
page[13]= "d=7-1/3 x 11 in|w=73|h=66|c=35mm-Slide"
```

C.3 Editing a QC7 File

The QC7 file (query configuration file) contains information about table relationships, security, export formats, label formats, page formats, etc. for a Results application. When you start Results with a database or set of databases for the first time, Results builds a QC7 file and names the file using the name of the first database connected (*dbname*`.qc7`).

By default, Results tries to create the QC7 file in the same directory that contains the first database connected. However, if the database is on another machine, Results might not be able to reach that directory. In that case, Results places the QC7 file in the current directory. Once Results creates the QC7 file, you can move it to any directory in the `PROPATH`. When you start Results, it first searches the current directory and then the directories of the `PROPATH` for a QC7 file. For Windows environments, the PROPATH environment variable is specified in the [Startup] section of the environment (.ini) file.

For more information about the PROPATH environment variable file, see the *Progress Installation and Configuration Guide Version 9 for UNIX* or *Progress Installation and Configuration Guide Version 9 for Windows*.

The Query→ Site Admin and Query→ Customize menus allow you to interactively edit a QC7 file. As a general rule, you should not edit a QC7 file using an ASCII editor. There are a few exceptions to this general rule:

- The r-code version has changed and you must regenerate the fastload system associated with the current QC7 file.
- The directory structure of your Results application has changed during the deployment process. If this happens, you must fix the pathnames specified for feature procedures, integration procedures, and icon files in the QC7 file prior to starting Results.

If you must edit a QC7 file with an ASCII editor, follow these guidelines:

- Always back up the QC7 file before editing it.
- Do not change the order of options before the fastload= line in the QC7 file. However, after the fastload= line, the order of entries after the QC7 file does not matter, as long as each group of entries is together.
- Delete the s.r file from the file. Deleting the s.r file from the QC7 file forces Results to regenerate the fastload system from the QC7 file the next time you start Results.

For more information about QC7 files, seeChapter 2, “Administering Results.”

C.4 Editing a QD7 File

A QD7 file (query directory file) contains query definitions. Results creates and maintains three distinct types of query directory files. The public query directory (`public.qd7`) contains query definitions accessible by all or a select group of users. Results also automatically generates a query directory for each user (*`userid`*`.qd7`) and uses the user ID as part of the query directory filename. For users who have the blank user ID, Results creates the `results.qd7` query directory file to store the query definitions of that user.

A single QD7 file can contain up to 256 query definitions. There is a query definition per line in a QD7 file and each definition has the following syntax:

SYNTAX

```
query[integerID]= "obsolete" "queryname" "dblist"
```

A query definition has a unique integer identifier, a name, and a comma-separated list containing the names of databases connected during the creation of the query. Leave the *obsolete* parameter empty.

Results uses the integer identifier to relate a query definition in a QD7 file with the query source file (`que#####.p`). For example, a query definition number 5 has a corresponding query source file named `que00005.p`. The query source files associated with the query definitions in a QD7 file must be located in the same operating system directory as the QD7 file.

The integer identifier for a query definition is unique across all query definitions in all QD7 files in a single operating system directory. An operating system directory can contain up to 99,999 query source files. Based on this limit and the fact that a QD7 file can contain up to 256 query definitions, do not place more than 390 QD7 files in a single operating system directory.

By default, Results creates QD7 files in the user's working directory. After Results creates a QD7 file, you can move the file to a directory in the `PROPATH`. If you move a QD7 file to a new directory, you must move all query source files associated with the QD7 file to the same directory. If another QD7 file already exists in the new directory, you might have to change the integer identifiers for query definitions and the name of associated query source files to ensure that queries are unique in the operating system directory.

See Chapter 2, "Administering Results," for more information about QD7 files.

C.5 Editing User Interface and Feature Files

The user interface (UI) file contains code that configures the tool bar and menu system for a Results application. The feature file contains code that defines and loads the features of a Results application. These two files are crucial to the deployment process of a Results application.

Results creates and maintains the UI and feature files. The Query→ Customize→ Deployment menu option allows you to rename these files. The Query→ Customize→ Menu Layout and Query→ Customize→ Tool Bar Layout menu options allow you to edit the UI file, and the Query→ Customize→ Feature menu option allows you to interactively edit the feature file. You can add and remove only custom features, not Results features.

The only time you should edit the UI and feature files using an ASCII editor is during the deployment process. If you intend to alter the directory structure of a Results application, you must fix the pathnames of application files prior to starting Results. Edit only the pathnames of the integration procedures and new feature procedures specified in the feature file. In the UI file, edit only the pathnames of icon files associated with new features added to the Results tool bar.

For more information about UI and feature files, see Chapter 3, “Programming Results.”

Index

Numbers

-1 (Single user Mode) startup parameter 5–9

A

AboutResults feature A–2

Accessing data 2–7

AdminCustomizeInterface feature A–2

AdminDeployment feature A–2

AdminExportType feature A–2

AdminFeatures feature A–3

AdminIntegration feature A–3

AdminLabelField feature A–3

AdminLabelType feature A–3

AdminMenuEdit feature A–4

AdminProgInitialize feature 3–25, A–4

AdminProgInstantiate feature 3–25, A–4

AdminProgWrite4GL feature 3–25, 3–50, A–5

AdminRebuild feature A–5

AdminReportType feature A–5

AdminReset feature A–6

AdminSecurity feature A–6

AdminSubMenu feature A–6

AdminTableAlias feature A–7

AdminTableRelations feature A–7

AdminToolbarEdit feature A–7

AdminWhere feature A–8

Aggregates
  defining A–28

Aliases
  query configuration files 2–6
  tables 2–5, A–7
    specifying 2–5

API. *See* Application Programming Interface

API shared variables
qbf-dtype 3–36
qbf-rct 3–35
qbf-rcw 3–35

Application
generated files C–2

Application databases. *See* Databases

Application Programming Interface 3–11, 3–20
basics 3–20
calculated fields 3–31
current view 3–26
editing queries 3–26
error messages 3–25
feature execution 3–24
field information 3–30
procedures 3–23
product version number 3–25
query fields 3–30
query names 3–27
query table list
accessing 3–27
setting 3–40
report formatting 3–36
sort order 3–29
table information
accessing 3–28

Application security 2–16, 2–17
features 2–17
integration points 3–7
QD7 files 2–19

Applications
base files 3–4
compiling 4–9
configuration 3–3
converting 5–8
databases. *See* Database
deployment process 4–2
deployment report 4–3
development process 3–2
directory structure 4–7
exit behavior 3–6
files 4–2, 4–6
generating code 3–50
icons 3–6, A–2
integrating Results 3–48
preferences 3–5
product name 3–5
PROPATH 4–8
QC7 files 2–3
rebuilding 2–9, 2–13
tool bar icons 4–4
window title bar A–2

Arrays A–8

Ask at Run Time 3–50, A–23

Audience xiii

B

Base files 3–4
fastload files 3–4
feature file 3–4
UI files 3–4

BEGINS operator 2–23

Bitmap images
source files 3–47

Blank user IDs 2–15

Bold typeface
as typographical convention xv

Browse view A–8

Browse views
queries A–18

C

CalcSubMenu feature A–8

Calculations
percent total A–20
running total A–25
string function A–27

Characters
delimiters 2–4, 3–6, A–11, C–4

Clipboard
output A–20

Closing
queries A–12

Code generation 3–49

Color settings
converting 5–7

Commands
PROSERVE 1–6
WPROSERVE 1–6

Configuration files
database 2–3
initial builds 2–3
query (QC7). *See* Query configuration files
query log files 2–4

configurationFileAltSrc 4–4
featureFile 4–4
menuFile 4–4

Configuring
applications 3–3

Connecting databases 1–5, 2–3

CONTAINS operator 2–23

Converter. *See* Results Converter

Counter A–9

CURRENT-WINDOW session handle 3–12

Custom relationships 2–9

Customize menu 3–3

Customizing
features 3–2, 3–20
help system 3–46
integration points 3–2
tool bar 3–18
user interface 3–8, 3–9

CustomLabel feature A–9

CustomReport feature A–9

D

Data
access
restricting 2–7
Data Administration tool 2–20
export formats A–26
exporting 3–45
security 2–20
features 2–24
integration points 3–7
table data selection 2–21
selection 2–21
selection criteria A–26
width A–13

Data Administration tool 2–15, 2–20

Data security 2–16

Database configuration files
initial builds. *See* Configuration files

Databases 2–2
accessing 1–4
connecting 2–3
connections 1–5
parameters 1–5, 5–9
deploying 4–4
deployment 4–7
design 3–2

multi-user 1–5, 5–9
schema 2–13
servers
shutting down 1–6
starting 1–6
single-user 1–5, 5–9

DataGovernor feature A–9

DateFunc feature A–10

-db (Physical Database Name)
startup parameter 5–9

-db startup parameter 1–5

DEFINED SHARED VARIABLE
statement 3–22

Deleting
queries A–12
relationships 2–13

Delimiters 3–6, A–11, C–4
characters 2–4

Deploying
help system 4–8

Deploying applications 4–2
application files 4–6
compiling 4–9
databases 4–4
development environment 4–2
directory structure 4–7
feature file 4–8
QC7 files 4–8
UI files 4–8
fastload system 4–9
feature file 4–4
feature procedures 4–5
files 4–2
help system 4–6
integration procedures 4–5
product icon file 4–5
PROPATH 4–8
QC7 files 4–4
QD7 files 4–6
query source files 4–6
report 4–3
results.l file 4–4
site requirements 4–2
tool bar icon files 4–4
u-export.p file 4–6
UI files 4–4

Deployment
user environment 4–2

Developing applications 3–2

Development environment
deployment process 4–2

E

Editing
feature files C–11
features 3–13
menus 3–15
QC7 files C–9
QD7 files C–10
results.l C–4
site defaults 2–25
table aliases 2–5
table relationships 2–7
tool bar 3–18
UI files C–11

Environment variables 4–8
PROPATH 1–5, 2–3, 3–5

Error messages A–17, A–23
API 3–25
displaying descriptions xix
QL files 2–14

Exit A–10

Exiting
Results 3–6

Export formats 2–26, C–5

Export procedure 3–45

EXPORT statement 3–50

Export view A–26
  queries A–18
  standard export formats 2–26

Export views
  field limits B–1

export.dat file 3–50

Exporting
  data 3–45
    delimiters A–11
    field lengths A–13
    separators A–12
  formats 2–4, 3–39, A–26
  records A–24
  setting formats A–2

ExportSubMenu feature A–10

ExportView feature A–11

F

Fastload system 3–4
  building 4–9
  deploying 4–9
  PROPATH 3–5

Feature Editor 3–13, A–3

Feature files 3–4
  deploying 4–4
  directory structure 4–8
  editing C–11

Feature procedures 3–10, 3–11
  arguments 3–14
  deploying 4–5
  executing features 3–24
  input parameters 3–11
  output parameters 3–11
  templates 3–12

Feature Security
  integration point 3–8

Features 3–2, A–1
  customizing A–3
  data security 2–24
  default menu label 3–14
  default tool bar icons 3–14
  defining 3–10, 3–13
  executing 3–24
  management 3–20
  menu option 3–15
  reset 3–20
  security 2–17
  sffire.p file 3–24
  toolbar icon 3–18
  user-defined A–6

Field security
  integration points 3–8

FieldDelimiters feature A–11

FieldProperties feature A–11

Fields 3–30, A–11
  aggregates A–28
  counter A–9
  delimiter characters A–11
  labels 2–28
  length A–13
  limits B–1
  logical A–16
  lookup A–17
  manipulating A–11
  numeric A–17, A–19
  properties A–11
  query information 3–30
  separators A–12
  sorting A–26
  summarizing A–28
  totals A–28

FieldsSeparators feature A–12

FileClose feature A–12

FileDelete feature A–12

FileOpen feature A–12

Files
base 3–4
converting 5–3, 5–11
database configuration. *See* Database configuration files
deploying 4–3
export.dat 3–50
feature 3–4
label.dat 3–50
naming
fastload A–2
feature definition A–2
GUI configuration A–2
public.qd7 2–15
query configuration
rebuilding A–5
See also Query configuration files
query directory. *See* Query directory files
Results C–1
results.l 2–4, 3–45, C–4
results.qd7 2–15
view
converting 5–7

FileSave feature A–13

FileSaveAs feature A–13

FixedWidthFields feature A–13

Footers
manipulating A–14

Form views
field limits B–1
queries A–18

FormView feature A–14

4GL code
generating A–5

FrameProperties feature A–14

Frames
properties A–14

G

Generating A–14
code 3–49
queries 3–40, 3–52

Graphics
conversion 4–6

H

Headers
manipulating A–14, A–20

HeadersAndFooters feature A–14

Help
Progress messages xix

Help systems
building 4–8
contents A–15
deployment 4–6, 4–8
how to use A–15
search A–15

Help viewer A–15

HelpContents feature A–15

HelpSearch feature A–15

HelpUse feature A–15

I

Icons 3–6
adding to tool bar 3–18
conversion 4–6
deploying 4–4
security 2–17
tool bar B–3

Includes A–15

Indexes
word 2–23

Information A–16

Initial builds
  QC7 files 2–3
  QL files 2–4
  results.l file 2–4

Integrating Results 3–48

Integration points 1–3, 3–2
  Feature Security 3–8
  Field Security 3–8
  Initialize Vars and Interface 3–8
  login program 3–9
  Menu Item Override 3–9, 3–19
  Table Security 3–9
  Table Selection Security 3–9

Integration procedures 1–3, 3–2, 3–7
  deploying 4–5

Italic typeface
  as typographical convention xv

J

Joins A–16
  See also Relationships
  self-joins 2–5

K

Keystrokes xv

L

-l startup parameter 1–5

Label view A–16, A–27
  maximum lines B–1
  queries A–19

Label views
  field selection behavior 2–28
  standard label sizes 2–26

label.dat file 3–50

Labels
  default selection A–3
  field information
    converting 5–5
  fields 2–28
  formats 3–40, A–27, C–7
  maximum lines B–1
  sizes 2–4, 2–26, A–3

LabelView feature A–16

Limits B–1

LIST operator 2–23

Logical values A–16

LogicalFunc feature A–16

Login Program
  integration points 3–9

Logo Screen
  integration point 3–9

Lookup A–17

M

Manual
  syntax notation xvi

Manual, organization of xiv

MasterDetail feature A–17

MATCHES operator 2–23

Math A–17

Maximum Memory (-mmax) startup parameter 1–5

Memory configuration 1–5

Menu Item Override integration point 3–9, 3–19

Menu options security 2–17

Menus
adding features 3–15
customizing A–4
display states 3–16
features 3–15
management 3–9, 3–16
reset 3–20
sensitivities 3–16

Messages A–17
displaying descriptions xix

-mmax startup parameter 1–5

Module permissions converting 5–5

Monospaced typeface as typographical convention xv

MPROWIN command 1–5

Multi-table queries 2–7

N

NewBrowseView feature A–18

NewDuplicateView feature A–18

NewExportView feature A–18

NewFormView feature A–18

NewLabelView feature A–19

NewReportView feature A–19

NewSubMenu feature A–19

NumericFunc feature A–19

O

OF relationships 2–7

Opening queries A–12

Operating system security 2–14

Operators
BEGINS 2–23
CONTAINS 2–23
LIST 2–23
MATCHES 2–23
RANGE 2–23

Options adding to menus 3–15

Output 3–43 procedures 3–44

OUTPUT statement 3–49

OUTPUT THROUGH statement 3–44

OUTPUT TO statement 3–44

OutputHeaderRecord feature A–20

P

-p (Startup Procedure) startup parameter 5–9

PageBreak feature A–20

Pages
customizing A–9
formats C–8
sizes 2–4, 2–25, A–5, A–27

Parameters
Physical Database Name (-db) 5–9
Single-user Mode (-1) 5–9
Startup procedure (-p) 5–9

Passwords. *See* Security

PercentTotal feature A–20

Physical Database Name (-db) startup parameter 1–5, 5–9

Preferences A–2
applications 3–5

PrintClip feature A–20

Printer information
converting 5–4

PrintFile feature A–21

Printing 3–43

PrintPreview feature A–21

PrintPrinter feature A–21

PrintPrinterNoBox feature A–22

PrintSubMenu feature A–22

Procedures
API 3–23
feature 3–10
initializing variables A–4
integrating A–3
integration 1–3, 3–2, 3–7
parameters 3–23
results.p 1–5, 3–48
sffire.p 3–23, 3–24
u-cprint.p 3–44
u-define.p 3–8
u-direct.p 3–9
u-export.p 3–45, 3–46
u-export.p file 4–6
u-field.p 3–8
u-login.p 3–9
u-logo.p 3–9
u-mcheck.p 3–9, 3–16
u-pprog.p 3–44
u-pview.p 3–44
u-secure.p 3–8
u-table.p 3–9
u-thru.p 3–44
u-to.p 3–44
u-where.p 3–9
vgrpti.p 3–23, 3–36
vgtbli.p 3–23, 3–28
vgtbll.p 3–23, 3–27
vsexpi.p 3–23, 3–39
vslbli.p 3–23, 3–40
vsrpti.p 3–23, 3–39
vstbli.p 3–23, 3–37
vstbll.p 3–23, 3–38, 3–40

Processing points 3–2

Product architecture 1–2

Product icons 3–6
deploying 4–5

Product name 3–5

ProgrammingSubMenu feature A–22

Progress
error messages 2–14, A–17, A–23
security 2–15, 2–20

Prompts 3–50

PROPATH
directory structure 4–8
fastload system 3–5
QC7 files 2–3
QD7 files 2–15
u-export.p file 3–46

PROPATH environment file 1–5

Properties
fields A–11
frames A–14

PROSERVE command 1–6

PROSHUT command 1–6

PROWIN command 1–5

_PROWIN executable 1–5

public.qd7 file 2–15

PUT CONTROL statement 3–44

PUT statement 3–50

Q

qbf-count shared variable 3–21

qbf-dtype shared variable 3–22, 3–36

qbf-governor shared variable 3–21

qbf-module shared variable 3–21, 3–26

qbf-name shared variable 3–21, 3–27

qbf-rc# shared variable 3–21, 3–30

qbf-rcc shared variable 3–21, 3–30

qbf-rcf shared variable 3–22, 3–34

qbf-rcg shared variable 3–21, 3–33

qbf-rcl shared variable 3–22, 3–34

qbf-rcn shared variable 3–21, 3–30

qbf-rcp shared variable 3–22, 3–34

qbf-rct shared variable 3–22, 3–35

qbf-rcw shared variable 3–22, 3–35

qbf-sortby shared variable 3–21, 3–29

qbf-vers shared variable 3–21, 3–25

QC7 files
query definition limits B–2
See also Query configuration files

QD7 files. *See* Query directory files

QL files. *See* Query configuration log files

Queries
aggregates 3–33
browse view A–8
calculated fields 3–30
closing A–12
converting 5–7
creating 3–40
dates A–10
defining A–18, A–19, A–26
definitions 2–15
deleting A–12
displaying A–11, A–18, A–19, A–25
displaying information A–16
export format 3–39
fields 3–30
counter A–9
data types 3–35
formats 3–34
labels 3–34
names 3–30
number of 3–30
width 3–35
form view A–14
frames A–14
function permissions
converting 5–5
generating 3–52
information 3–21
inserting code A–15
label format 3–40
label view A–9
limiting records A–9
lookup fields A–17

multiple tables 2–7
name 3–27
opening A–12
previewing output A–21
printing output A–21, A–22
report format 3–39
saving A–13
sending output to clipboard A–20
sending output to file A–21
sort fields 3–29
source code files A–14
table information 3–37
accessing 3–28
table list 3–38
accessing 3–27
setting 3–40
tables 3–27, A–28
view 3–26
view layout 3–34

Query
name length limit B–2

Query configuration files
aliases 2–6
converting 5–11, 5–12
database connections 2–3
deploying 4–4
directory structure 4–8
editing 3–6, C–9
fastload system 3–4
feature security 2–18
inital builds 2–3
location 2–3
multiple databases 2–3
PROPATH 2–3
rebuilding 2–13, A–5
results.l 2–4
save behavior 3–6
standard export formats 2–26
standard label sizes 2–26
standard page sizes 2–25
table relationships 2–7, 2–8

Query configuration log files 2–4
rebuild messages 2–14

Query definition
limits B–2

Query directories 2–15
security 2–16, 2–19, A–23, A–29

Query directory files 2–15
blank user 2–15
creating 2–15
deployment 4–6
editing C–10
integration points 3–9
location 2–15
public 2–15
user IDs 2–15

Query files
converting 5–12

Query source file
deployment 4–6
limits B–2

R

RANGE operator 2–23

ReadOtherDirectory feature 2–16, 2–19, A–23

ReadPublicDirectory feature 2–16, 2–19, A–23

ReAsk feature A–23

Rebuilding
applications 2–9
QC7 files 2–13
queries 2–13
table relationships 2–13

RecentMessages feature A–23

RecordAdd feature 2–24, A–24

RecordDelete feature 2–24, A–24

RecordEnd feature A–24

Records
exporting A–24
limiting A–9
security A–24, A–25

RecordStart feature A–24

RecordUpdate feature 2–24, A–25

Relationships 2–7
  building 2–12
  converting 5–6
  creating 2–9
  custom 2–9
  defining A–16
  deleting 2–13
  editing 2–7, 2–10
  OF relationships 2–7
  rebuilding 2–13
  tables A–7
  WHERE relationships 2–11

Report format 3–39

Report layout
  converting 5–6

ReportView feature A–25

Report views A–25, A–27
  field limits B–1
  formatting 3–36
  queries A–19
  standard page sizes 2–25

Reports
  default format 2–4

reseng.hpj file 3–46

reseng.rft file 3–47

reseng2.rft file 3–47

result.l file
  deploying 4–4

Results
  add calculated field menu A–8
  administration menu A–6
  API. *See* Application Programming Interface
  architecture 1–2
  code generation 3–49
  custom export menu A–10
  customize menu A–22
  customizing 1–3
  delimiter characters 3–6
  deploying 4–2
  exit behavior A–2
  exiting 3–6, A–10
  exporting data 3–45
  features A–1
    customizing A–3
  files C–1
  generated files C–2
  help system A–15
    source files 3–46
  installing A–2
  integrating 3–48
  limits B–1
  main procedure 5–9
  memory configuration 1–5
  menu bar
    customizing A–6
  menu system 3–15
  new query menu A–19
  output solutions 3–43
  print menu A–22
  product directory 1–5
  programmer 1–4
  query configuration files 2–3
  security A–6
  site administrator 1–4
  starting 1–4
  startup procedure 3–48, 5–9
  stopping 1–4
  tool bar 3–18
    customizing A–6
  version 2 1–1

Results Converter 5–1
  applications 5–8
  color settings 5–7
  files 5–3
  label field information 5–5
  module permissions 5–5
  printer information 5–4
  queries 5–7
  query function permissions 5–5
  relationships 5–6
  report layout 5–6

results.l file 5–8
starting 5–8
user-added code 5–8
view files 5–7

Results engine 1–3

results.l
delimiters C–4
editing C–4
standard export formats C–5
standard label sizes C–7
standard page sizes C–8

results.l file 2–4
converting 5–8
standard export formats 2–26, 3–45
standard label sizes 2–26
standard page sizes 2–25

results.p file 1–5, 3–48, 5–9

results.qd7 file 2–15

RUN statement 3–23

Run time
user input A–23

RunningTotal feature A–25

S

-s startup parameter B–2

Saving
queries A–13

Security
application 2–16, 2–17
data 2–16, 2–20
features 2–17
integration points 3–7
QD7 files 2–16, 2–19
query directory files 2–15
tables 3–9
user access 2–16, A–23, A–24, A–25, A–29
user IDs 2–14

Selection A–26

Selection criteria
API procedures
vgtbli.p file 3–28
BEGINS operator 2–23
CONTAINS operator 2–23
defining 2–24
integration points 3–9
limits B–2
LIST operator 2–23
MATCHES operator 2–23
RANGE operator 2–23
run-time prompts 2–24
tables A–8
word indexes 2–23

Self-joins 2–5

Server shutdown command 1–6

Servers
starting 1–6

Session handles
CURRENT-WINDOW 3–12

sffire.p file 3–23
parameters 3–24

Shared variables
API 3–21
qbf-count 3–21
qbf-dtype 3–22
qbf-governor 3–21
qbf-module 3–21, 3–26
qbf-name 3–21, 3–27
qbf-rc# 3–21, 3–30
qbf-rcc 3–21, 3–30
qbf-rcf 3–22, 3–34
qbf-rcg 3–21, 3–33
qbf-rcl 3–22, 3–34
qbf-rcn 3–21, 3–30
qbf-rcp 3–22, 3–34
qbf-rct 3–22
qbf-rcw 3–22
qbf-sortby 3–21, 3–29
qbf-vers 3–21, 3–25
u-pvars.i 3–22

Shutdown commands 1–4

Single-user Mode (-1) startup parameter 1–5, 5–9

Site administration
  menu 2–2

Site defaults 2–4, 2–25
  editing 2–4
  label field selection 2–28
  standard export formats 2–26
  standard label sizes 2–26
  standard page sizes 2–25

Sorting data A–26

SortOrdering feature A–26

Source files
  bitmaps 3–47
  reseng.hpj 3–46
  reseng.rtf 3–47
  reseng2.rtf 3–47

Stack Size (-s) startup parameter B–2

Stacked arrays A–8

Standard export formats 2–26

Standard label sizes 2–26

Standard page sizes 2–25

StandardExport feature A–26

StandardLabel feature A–27

StandardReport feature A–27

Starting
  Results Converter 5–8

Starting Results 1–4

Startup commands 1–4
  MPROWIN 1–5
  PROWIN 1–5

Startup parameters
  Maximum Memory (-mmax) 1–5
  Physical Database Name (-db) startup parameter 1–5
  Single-user Mode (-1) 1–5
  Stack Size (-s) B–2
  Stash Area (-stsh) B–2

Startup Procedure (-p) startup parameter 1–5, 5–9

Stash Area (-stsh) startup parameter B–2

Statements
  DEFINE SHARED VARIABLE 3–22
  EXPORT 3–50
  OUTPUT THROUGH 3–44
  OUTPUT TO 3–44
  PUT 3–50
  PUT CONTROL 3–44
  RUN 3–23
  TERMINAL PAGED 3–49
  UPDATE 3–51
  WAIT-FOR 3–49

Status line
  displaying A–27

StatusLine feature A–27

Stopping Results 1–4

StringFunc feature A–27

-stsh startup parameter B–2

Syntax notation xvi

T

Table
  sort field limit B–2

Table aliases
  creating 2–6
  specifying 2–5

Table relationships
building 2–12
creating 2–9
custom 2–9
default 2–7
deleting 2–13
editing 2–7, 2–10
OF relationships 2–7
rebuilding 2–13
WHERE relationships 2–11
limits B–2

Table Selection Security
integration point 3–9

Tables A–28
aliases 2–5, A–7
break points A–17
data selection 2–21
including in queries A–15
information
accessing 3–28
joining 2–5
queries 3–37, 3–38, A–28
query information 3–27
relationships 2–7, A–7
security 3–9
selection criteria 2–21, A–8

Templates
feature procedures 3–12

TERMINAL PAGED statement
3–49

Tool bar A–28
customizing A–7
displaying A–28
editing 3–18
features 3–18
icon display state 3–19
icon sensitivity 3–19
icons 3–19
limits B–3
management 3–19
managing 3–9, 3–19
reset 3–20

Totals A–28

TotalsOnly feature A–28

Typographical conventions xv

U

u-cprint.p file 3–44

u-define.p file 3–8

u-direct.p file 3–9

u-export.p file 3–45
deployment 4–6
PROPATH 3–46

u-field.p file 3–8

u-login.p file 3–9

u-logo.p file 3–9

u-mcheck.p file 3–9, 3–16

u-pprog.p file 3–44

u-pvars.i file 3–22

u-pview.p file 3–44

u-secure.p file 3–8

u-table.p file 3–9

u-thru.p file 3–44

u-to.p file 3–44

u-where.p file 3–9

UI files 3–4
deploying 4–4
directory structure 4–8
editing C–11

UPDATE statement 3–51

User environment
deployment process 4–2
requirements 4–2

User IDs 2–14
  blank 2–15
  feature security 2–18
  lists 2–18
  operating system 2–14
  passwords 2–14
  Progress 2–15
  QD7 files 2–15
  specifications 2–18

User interface
  integration points 3–8
  management 3–20
  menu management 3–9, 3–16
  product icons 3–6
  product name 3–5
  tool bar management 3–9, 3–19

User lists 2–18

User-added code
  converting 5–8

Users
  access A–6

V

Variables
  API 3–21
  initializing A–4

vgrpti.p file 3–23
  parameters 3–36

vgtbli.p file 3–23
  parameters 3–28

vgtbll.p file 3–23
  parameters 3–27

View files
  converting 5–7

Viewer
  help A–15

Views 3–26
  browse A–8, A–18
  export A–11, A–18, A–26
  form A–14, A–18
  label A–9, A–16, A–19, A–27
  report A–9, A–19, A–27

vsexpi.p file 3–23
  parameters 3–39

vslbli.p file 3–23
  parameters 3–40

vsrpti.p file 3–23
  parameters 3–39

vstbli.p file 3–23
  parameters 3–37

vstbll.p file 3–23
  parameters 3–38, 3–40

W

WAIT-FOR statement 3–49

WHERE clause 3–9, A–26, B–2
  table relationships 2–11
  tables A–8

Word indexes 2–23

WPROSERV executable 1–6

WPROSERVE command 1–6

WPROSHUT executable 1–6

WriteOtherDirectory feature 2–16, 2–19, A–29

WritePublicDirectory feature 2–16, 2–19, A–29