

Progress
Installation and Configuration Guide
Version 9
for UNIX

© 2002 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, Powered by Progress, Progress Fast Track, Progress Profiles, Partners in Progress, Partners en Progress, Progress en Partners, Progress in Progress, P.I.P., Progress Results, ProVision, ProCare, ProtoSpeed, SmartBeans, SpeedScript, and WebSpeed are registered trademarks of Progress Software Corporation in the U.S. and other countries. A Data Center of Your Very Own, Allegrix, Apptivity, AppsAlive, AppServer, ASPen, ASP-in-a-Box, Empowerment Center, Fathom, Future Proof, IntelliStream, OpenEdge, PeerDirect, POSSE, POSSENET, Progress Dynamics, Progress Software Developers Network, SectorAlliance, SmartObjects and WebClient are trademarks or service marks of Progress Software Corporation in the U.S. and other countries.

SonicMQ is a registered trademark of Sonic Software Corporation in the U.S. and other countries.

Vermont Views is a registered trademark of Vermont Creative Software in the U.S. and other countries.

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

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

Progress Software Corporation acknowledges the use of Raster Imaging Technology copyrighted by Snowbound Software 1993-1997, the IBM XML Parser for Java Edition, and software developed by the Apache Software Foundation (http://www.apache.org/).

© 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.

The implementation of the MD5 Message-Digest Algorithm used by Progress Software Corporation in certain of its products is derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm.

May 2002



Product Code: 4500
Item Number: 89427U;V91D

Contents

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

PART I Installation

1. Installation Requirements **1–1**
1.1 System Requirements 1–2
1.1.1 Java 1–2
1.1.2 Requirements For Building Version 9 Applications 1–6
1.1.3 Requirements For Running Version 9 Applications 1–6
1.1.4 Disk Space Requirements 1–6

1.2 4GL Client/Database Server Compatibility 1–8
1.3 Supported Platforms 1–9
1.4 Licensing 1–9
1.5 Product Life Cycle 1–10

2. Installing Progress 2–1
2.1 Performing Preinstallation Tasks 2–2
2.1.1 Reading the Release Notes 2–2
2.1.2 Installing Other Required Applications 2–2
2.1.3 Upgrading an Existing Progress Installation 2–3
2.2 Beginning the Installation 2–4
2.3 Creating a Directory For a First-time Progress Version 9.1 Installation . 2–6
2.4 The JavaHome Path 2–6
2.5 Installing Progress On Your System 2–6
2.6 Performing a Typical Or Complete Installation 2–14
2.7 Performing a Custom Installation 2–21
2.8 Running the Installation Utility In Batch Mode 2–30
2.8.1 Support Files 2–30
2.8.2 Installation .ini File 2–30
2.8.3 Batch Installation Using the -b Switch 2–33
2.8.4 Log File 2–33
2.9 Performing Postinstallation Tasks 2–34
2.9.1 Required Tasks 2–34
2.9.2 Setting AdminServer Security 2–34
2.10 Downloading Executables For Heterogeneous Environments 2–35
2.11 Installing Additional Products 2–36
2.11.1 Changes In the Progress Core Installation Program For Progress Messages Files 2–37
2.12 Adding Components To Previously Installed Products 2–43

3. System Administration Utilities 3–1
3.1 Using the Product Update Utility 3–2
3.2 Using the SHOWCFG Utility 3–3
3.3 Managing User Licenses 3–5
3.3.1 Progress License Information 3–5
3.3.2 Using the Progress License File 3–5
3.4 Progress Event Logging 3–8
3.4.1 Progress Event Log File 3–8
3.4.2 Managing the Progress Event Log File Size 3–8

4. Troubleshooting 4–1
4.1 Error Messages 4–2
4.2 Altered Or Missing progress.cfg File 4–2
4.3 Tailoring Startup Scripts 4–3

PART II Configuration

5. Setting Up Your Environment 5–1
5.1 Setting Environment Variables 5–2
5.2 UNIX Environment Variables 5–2
5.3 Setting Java Environment Variables 5–7
5.3.1 Setting the JDK Environment Variable 5–8
5.4 Setting SQL Client Environment Variables 5–8
5.5 Using the Open Client Toolkit and the Just-in-Time Compiler 5–9
5.6 Running Progress Applications 5–9
5.7 Using the Proenv Utility 5–10
5.8 Using the AdminServer 5–10
5.8.1 proadsv 5–11
5.8.2 How To Implement the User-Group Authorization Feature . . . 5–13

6. Memory and System Configuration 6–1
6.1 Calculating Memory Needs 6–2
6.2 Managing Shared Memory and Process Resources 6–6
6.2.1 Shared Memory 6–6
6.2.2 Processes 6–7
6.3 Reducing Memory Usage 6–7
6.4 Swap Space 6–8
6.5 Shared Memory and Kernel Configuration 6–8

7. Configuring Progress 7–1
7.1 The Progress Explorer Framework 7–2
7.1.1 AdminServer 7–2
7.1.2 Command-line Configuration Utilities 7–2
7.1.3 Managing Progress Product Configurations 7–4
7.2 Shared-memory Configurations 7–7
7.3 Multi-tier Configurations 7–10
7.3.1 Shared-memory Architecture 7–11
7.4 Client/Server Configurations 7–13
7.4.1 Terminology 7–13
7.4.2 Simple Client/Server Configurations 7–15
7.5 Client/Server and Progress AppServer In the Network Environment . . . 7–16
7.6 Preparing To Run Progress In a TCP/IP Network 7–23
7.6.1 Configuring UNIX Machines In a TCP/IP Network 7–23
7.6.2 Installing Progress On Your TCP/IP Network 7–24
7.6.3 Typical TCP/IP Configuration With a Hard Disk On Each Machine 7–25
7.6.4 Setting Up Network Files To Run Progress 7–26
7.6.5 Configuring Progress On a Network Operating System. 7–26

8. Configuring Progress Unified Broker Products 8–1
8.1 The Progress Unified Broker Administration Framework 8–2
8.1.1 Unified Broker Clients 8–4
8.1.2 Progress Explorer and Management Utilities 8–4
8.1.3 NameServers 8–7
8.1.4 Unified Broker 8–8
8.1.5 Unified Broker Properties File (ubroker.properties). 8–10
8.1.6 Using a Text Editor and Configuration Utilities 8–11
8.1.7 Overview Of Unified Broker Administration 8–13
8.1.8 Distributing Machines For a Unified Broker Product 8–16
8.1.9 Configuring NameServer Communications. 8–17
8.2 Configuring Progress Unified Broker Components 8–18
8.2.1 Downloading NameServer Executables 8–19
8.2.2 Order Of Configuration 8–19
8.2.3 Configuring and Using NameServer Instances. 8–19
8.2.4 Editing the Properties File. 8–20
8.3 Setting Up the Progress Unified Broker Environment 8–23
8.3.1 Setting Environment Variables 8–23
8.3.2 Configuring Progress Unified Broker Log Files 8–24
8.3.3 Other Preparations Prior To Startup 8–25
8.4 Starting Progress Unified Brokers 8–25
8.4.1 Using the Management Utilities To Start a Unified Broker Instance. 8–25
8.4.2 Starting and Managing a NameServer Using the NSMAN Utility 8–26
8.5 Understanding Load Balancing 8–29
8.6 Understanding Server-level and Connection-level Fault Tolerance 8–31
8.6.1 Connection-level Fault Tolerance 8–32
8.6.2 Using UDP Broadcasting 8–33
8.6.3 Using NameServer Replication. 8–33
8.6.4 Using NameServer Neighbors 8–36
8.6.5 Performance Implications Of Broadcasting. 8–39

9. Maintaining Two Versions 9–1
9.1 Maintaining Two Versions 9–2
9.2 Running Multiple AdminServers 9–3

10. Starting and Running Progress 10–1
10.1 Starting Progress 10–2
10.1.1 Startup and Shutdown Commands 10–2
10.1.2 Starting Single-user Progress In Interactive Mode 10–4
10.1.3 Starting Single-user Progress In Batch Or Background Mode . 10–5
10.1.4 Starting the Multi-user Server Or Broker. 10–6

10.2 Running Progress Clients and Servers On a Network 10–6
10.2.1 Using Network Startup Parameters 10–6
10.2.2 Specifying the Network Type (-N) 10–8
10.2.3 Network Addressing With Progress (-S and -H) 10–9
10.2.4 Starting Applications On a Network 10–9
10.2.5 Starting Multiple Brokers Using the Same Protocol 10–10
10.2.6 Accessing a Server Behind a Firewall 10–11
10.2.7 Starting and Running Multi-user Progress In Interactive Mode 10–12
10.2.8 Starting and Running Multi-user Progress Clients In Batch Or Background Mode 10–13

A. Terminals A–1
A.1 Terminal Issues A–2
A.2 Terminal Identifiers A–2

B. Progress National Language Support B–1
B.1 Packaging B–2
B.2 Directory Structure B–3
B.3 Contents Of Each Directory B–3
B.4 Implementing Regional Support B–3
B.5 International Databases B–4
B.6 Progress Messages B–4
B.7 Environment Variables Of the SQL-92 Client B–5
B.8 Regional Parameter Files B–6

C. The Progress Directory Structure C–1
C.1 Installation Path Directory Structure C–2

D. Progress Version 9 Typical and Complete Installation Components D–1
D.1 Progress Version 9 Installation Options D–2
D.2 Progress Product Components and Subcomponents D–2
D.2.1 4GL Development D–2
D.2.2 AppServer Internet Adapter D–6
D.2.3 AppServer Internet Adapter/S D–7
D.2.4 Client Networking D–7
D.2.5 Client-side Security D–9
D.2.6 Debugger D–10
D.2.7 Enterprise Database D–10
D.2.8 ESQL-89/C D–12
D.2.9 NameServer D–12
D.2.10 NameServer Load Balancing D–13
D.2.11 Oracle DataServer D–13

D.2.12 Personal Database D–15
D.2.13 Progress AppServer D–17
D.2.14 ProVision Development Server D–18
D.2.15 ProVision Plus Development Server D–22
D.2.16 Query/Results D–25
D.2.17 Secure AppServer D–26
D.2.18 SonicMQ Adapter D–28
D.2.19 SQL-92 Client Access D–30
D.2.20 Toolkit D–31
D.2.21 WebSpeed Development Server D–31
D.2.22 WebSpeed Enterprise Transaction Server D–34
D.2.23 WebSpeed Messenger D–37
D.2.24 WebSpeed Transaction Server D–37
D.2.25 Workgroup Database D–39

Index Index–1

Figures

Figure 3–1: Product Configuration Displayed Using SHOWCFG Utility 3–4
Figure 7–1: Database Access On a Shared-memory System 7–9
Figure 7–2: Sample Multi-tier Configuration 7–10
Figure 7–3: Shared-memory Progress Architecture 7–11
Figure 7–4: Simple Client/Server Configuration 7–15
Figure 7–5: Multiple System Client/Server Configuration 7–16
Figure 7–6: Simple Progress Network Configuration 7–18
Figure 7–7: Network File Server For Application Files 7–19
Figure 7–8: Network File Server As a Database Server 7–20
Figure 7–9: Network File Server For Application and Database Files 7–21
Figure 7–10: LAN Configuration With the Progress AppServer 7–22
Figure 7–11: Typical TCP/IP Configuration (File Server Not Used) 7–25
Figure 7–12: Sample Network Resources 7–27
Figure 8–1: Progress Unified Broker Administration Framework 8–2
Figure 8–2: Progress Unified Broker Administration Overview 8–14
Figure 8–3: Maximum Unified Broker Machine Distribution 8–16
Figure 8–4: Sample Unified Broker Client Services File 8–18
Figure 8–5: Server-level and Connection-level Fault Tolerance 8–31
Figure 8–6: NameServer Replication 8–34
Figure 8–7: NameServer Neighbors 8–38

Tables

Table 1–1: JRE/JDK Requirements By Platform 1–4
Table 1–2: Requirements For Running Version 9 Applications 1–6
Table 1–3: Approximate Disk Space Requirements 1–7
Table 1–4: Client/Server Compatibility 1–8
Table 1–5: Supported Platforms 1–9
Table 1–6: Product Life Cycle Phases 1–10
Table 2–1: Installation Options 2–5
Table 2–2: Mount Commands 2–7
Table 2–3: Product Configuration Data Dialog Box Options 2–10
Table 2–4: Type, Device and Destination Dialog Box Options 2–13
Table 2–5: Destination Pathname Exists Dialog Box Options 2–15
Table 2–6: Custom - Select Products Dialog Box Status Information 2–23
Table 2–7: User-Group Parameter Options 2–35
Table 2–8: Promsgs File Synchronization Process 2–40
Table 2–9: Example Of Promsgs Files Being Out Of Sync 2–42
Table 4–1: Error Messages 4–2
Table 4–2: Reasons For Altered Or Missing progress.cfg File 4–2
Table 5–1: UNIX Environment Variables 5–3
Table 5–2: proadsv Command-line Options 5–12
Table 5–3: User-Group Parameter Options 5–13
Table 6–1: Components Used To Calculate Memory Needs 6–2
Table 6–2: Size Increments For Increasing Startup Parameters By 1 6–3
Table 6–3: Single-user Memory Requirements 6–4
Table 6–4: Multi-user Memory Requirements 6–4
Table 6–5: Formulas For Calculating Memory Requirements 6–5
Table 6–6: Shared Memory and Semaphore Parameter Settings 6–9
Table 6–7: Error Codes and Kernel Reconfiguration Parameters 6–10
Table 7–1: ubroker.properties File Validation Utilities 7–6
Table 7–2: ubroker.properties File Management Utilities 7–7
Table 7–3: Progress Network Type Options For Progress 7–17
Table 7–4: TCP/IP Network Files 7–26
Table 8–1: Unified Broker Administration Summary 8–3
Table 8–2: Weight Factors Based On Percentage 8–29
Table 8–3: Weight Factors Based On Arbitrary Sums 8–30
Table 10–1: Progress Command Components 10–2
Table 10–2: UNIX Startup and Shutdown Commands 10–3
Table 10–3: Client Parameters 10–7
Table 10–4: Server Parameters 10–7
Table 10–5: Default Network Types 10–8
Table A–1: Terminal Identifiers A–2
Table B–1: PROMSGS Translations Shipped With Progress B–2
Table B–2: National Language File Descriptions B–3
Table B–3: Startup Parameters For a Deployed Application B–6

Table C–1: Progress ($DLC) Directory Structure C–2
Table D–1: 4GL Development Components and Subcomponents D–3
Table D–2: AppServer Internet Adapter Components and Subcomponents D–7
Table D–3: AppServer Internet Adapter/S Components and Subcomponents D–7
Table D–4: Client Networking Components and Subcomponents D–8
Table D–5: Client-side Security Components and Subcomponents D–9
Table D–6: Debugger Components D–10
Table D–7: Enterprise Database Components and Subcomponents D–10
Table D–8: ESQL-89/C Components D–12
Table D–9: NameServer Components and Subcomponents D–12
Table D–10: NameServer Load Balancing Components and Subcomponents D–13
Table D–11: Oracle DataServer Components and Subcomponents D–13
Table D–12: Personal Database Components and Subcomponents D–15
Table D–13: Progress AppServer Components and Subcomponents D–17
Table D–14: ProVision Development Server Components and Subcomponents . . D–19
Table D–15: ProVision Plus Development Server Components and Subcomponents D–22
Table D–16: Query/Results Components and Subcomponents D–25
Table D–17: Secure AppServer Components and Subcomponents D–27
Table D–18: SonicMQ Adapter Components and Subcomponents D–28
Table D–19: SQL-92 Client Access Components and Subcomponents D–30
Table D–20: Toolkit Components D–31
Table D–21: WebSpeed Development Server Components and Subcomponents D–31
Table D–22: WebSpeed Enterprise Transaction Server Components and Subcomponents D–34
Table D–23: WebSpeed Messenger Components D–37
Table D–24: WebSpeed Transaction Server Components and Subcomponents . . D–37
Table D–25: Workgroup Database Components and Subcomponents D–40

Procedures

setup.ini 2–31

Preface

Purpose

This book describes how to install and set up Progress Version 9 for UNIX and Linux operating systems. It also provides terminal definitions and supplemental information.

Audience

This book is intended for the person installing Progress. You are not required to have prior knowledge of Progress, but you must have root privilege and be familiar with the UNIX or Linux operating system.

Organization Of This Manual

Part I, Installation

Chapter 1, “Installation Requirements”

Lists the system requirements for installing Progress products.

Chapter 2, “Installing Progress”

Provides step-by-step instructions for installing Progress products using a typical, a complete, or a custom installation from CD.

Chapter 3, “System Administration Utilities”

Provides information about the Product Update utility, the SHOWCFG utility, managing user licenses, the NSCONFIG utility, and Progress event logging.

Chapter 4, “Troubleshooting”

Lists error messages, their causes, and their solutions.

Part II, Configuration

Chapter 5, “Setting Up Your Environment”

Discusses setting Version 9.1 system and Java environment variables, managing the AdminServer, and running Progress Version 9.1.

Chapter 6, “Memory and System Configuration”

Describes how to calculate memory needs and suggests ways to conserve memory. This chapter also explains how to configure your system.

Chapter 7, “Configuring Progress”

Provides information about shared-memory configurations, multi-tier configurations, client/server configurations, and preparing to run Progress on a TCP/IP network.

Chapter 8, “Configuring Progress Unified Broker Products”

Describes the common framework and tasks to configure the common components of Progress Application Server products. These products all support multi-tier or client/server configurations using the Progress NameServer.

Chapter 9, “Maintaining Two Versions”

Provides information about maintaining Version 9 and either Version 7 or Version 8 of Progress on your system.

Chapter 10, “Starting and Running Progress”

Provides instructions for starting and running a Progress database in different modes. Also provides instructions for running Progress clients and servers on a network.

Appendix A, “Terminals”

Provides a list of built-in terminal definitions supplied with Progress.

Appendix B, “Progress National Language Support”

Provides information about Progress messages in different languages, including information about language packaging, directory structures and contents, regional support, international databases, and regional parameter files.

Appendix C, “The Progress Directory Structure”

Lists the subdirectories of your installation path.

Appendix D, “Progress Version 9 Typical and Complete Installation Components”

Provides a list of the components and subcomponents that you install for each product when you choose either a Typical or Complete installation.

Typographical Conventions

This manual uses the following typographical conventions:

- **Bold typeface** indicates:
  - Commands or characters that the user types
  - That a word carries particular weight or emphasis
- *Italic typeface* indicates:
  - Progress variable information that the user supplies
  - New terms
  - Titles of complete publications

- `Monospaced typeface` indicates:
  - Code examples
  - System output
  - Operating system filenames and pathnames

The following typographical conventions are used to represent keystrokes:

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

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

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

  CTRL-X

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

  ESCAPE H
  ESCAPE CURSOR-LEFT

Syntax Notation

The syntax for each component follows a set of conventions:

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

  In this example, ACCUM is a keyword:

  SYNTAX

  ```
  ACCUM aggregate expression
  ```

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

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

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

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

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

  SYNTAX

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

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

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

  SYNTAX

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

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

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

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

  SYNTAX

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

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

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

SYNTAX

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

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

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

  SYNTAX

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

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

  SYNTAX

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

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

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

  SYNTAX

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

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

  SYNTAX

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

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

SYNTAX

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

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

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

  SYNTAX

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

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

  SYNTAX

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

Progress Messages

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

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

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

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

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

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

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

1 ♦ Start the Progress Procedure Editor:

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

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

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

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

Other Useful Documentation

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

Getting Started

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

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

Progress Installation and Configuration Guide Version 9 for 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 bulletin that provides a list of new and changed features by release, a list and description of changes to documentation by release, and critical information about product changes that might require changes to existing code and configurations.

This bulletin also provides information about where to go for detailed information about the new and changed features and documentation.

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 and shutdown commands that you use at the command line, and the startup parameters that you use for Progress processes. This guide also provides information about parameter usage and parameter files.

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 Administration and Development Guide (Windows only)

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

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

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

4GL

Building Distributed Applications Using the Progress AppServer

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

Progress External Program Interfaces

A guide to the external programming interfaces supported by Progress. This manual covers the Host Language Call (HLC) Interface, the system clipboard, named pipes, shared libraries and DLLS, Windows Dynamic Data Exchange (DDE), COM objects, ActiveX Automation, ActiveX controls, sockets, XML, SAX, and the SonicMQ 4GL Adapter.

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 on the Web

A manual that describes how to use the new WebClient, AppServer Internet Adapter, SmartObjects, and SonicMQ Adapter to create applications tailored for Internet, intranet, and extranet environments.

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 as follows:

- *Progress/400 Product Guide*
- *Progress DataServer for Microsoft SQL Server Guide*
- *Progress DataServer for ODBC Guide*
- *Progress DataServer for ORACLE 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 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/products/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, build, and deploy Java and ActiveX applications, and Java applets that run as clients of the Progress AppServer. This guide includes information about how to expose the AppServer as a set of Java classes or as an ActiveX server, and how to choose an Open Client distribution package fro run time.

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.

PART I

Installation

Installation Requirements

Installing Progress

System Administration Utilities

Troubleshooting

1

Installation Requirements

This document tells you how to install Progress Version 9.1 on a machine running UNIX or Linux operating systems.

Specifically, this chapter contains the following sections:

- System Requirements
- 4GL Client/Database Server Compatibility
- Supported Platforms
- Licensing
- Product Life Cycle

1.1 System Requirements

This section describes the hardware and software requirements for running Progress Version 9.1 on UNIX and Linux.

NOTE: The system requirements provided in this chapter are current as of the publication date of this manual; however, requirements can change. To make sure that you have the most up-to-date information about system requirements, please be sure to refer to the *Progress Release Notes*.

1.1.1 Java

Many Version 9 products require one or more installed Java components for subsequent use by Progress. Specifically, the components are the Java Run-time Environment (JRE) and the Java Development Kit (JDK).

The JDK contains the software and tools that developers need to compile, debug, and run applets and applications written using the Java programming language. The JDK software and documentation, typically included with each new release of an operating system, are available for download at the vendor's Web site. You will require a JDK if you intend to develop Java stored procedures and triggers with the Database, or to create Java proxies with the Open Client Toolkit.

The JRE consists of the Java Virtual Machine, the Java Core Classes, and supporting files. The JRE is the run time part of the JDK and does not include a compiler, a debugger, or development tools. You will need the JRE if you intend to use one of the following:

- AppServer
- Java application or applet
- Progress Explorer
- SonicMQ Adapter
- SQL Explorer
- WebSpeed Transaction Server

You will also need the JRE to execute Java stored procedures and triggers from the database.

The installation program automatically installs the correct JRE if you install Progress 9.1 products on either of the following 32-bit platforms:

- Sun Solaris SPARC/Intel, Versions 2.6, 2.7, and 2.8
- HP-UX Version 11

Additionally, the following list identifies 64-bit platforms that install the 32-bit JRE:

- HP-UX Version 11, 64-bit HP-UX (installed automatically)
- SUN Solaris, Sol64
- SUN Solaris SPARC, Version 8

NOTE: If you are using the 32-bit JRE on a 64-bit machine, Java triggers and stored procedures are not supported.

On the Sun Solaris platform, the Progress Version 9.1 installation program also automatically installs the JDK when you install a Version 9 product that requires the JDK.

In some cases, you might need to install some required Java components, as described below:

- **HP-UX** — The installation program does not automatically install the JDK component if you install Progress on this platform. If you want access to the JDK and do not already have it installed, you should install the required version of the JDK on your target system. For more information on Java requirements, see Table 1–1.

  Additionally, it may be necessary for you to adjust the default kernel parameter max_thread_proc. To determine whether the default kernel parameter is too low and should be modified, contact your HP or system administrator.

- **Compaq Tru64, SCO OpenServer 5, and Linux platforms** — The installation program does not automatically install the JRE or JDK component if you install Progress component on one of these platforms. You must install the required version of the JRE and/or the JDK (if not already installed) on your target system. For more information on Java requirements, see Table 1–1.

- **SQL-92** — The installation program does not automatically install the JDK component when you install the Enterprise Database, the Personal Database, or the Workgroup Database with SQL Client Access. If you intend to develop Java stored procedures and Java triggers for your database, you must install a Progress development product such as ProVision. For information on writing Java stored procedures and triggers, see the *Progress SQL-92 Guide and Reference*.

  NOTE: The SQL-92 Server Engine is not available on SCO OpenServer 5 and Solaris Intel platforms.

Table 1–1 lists operating systems and specifies the versions of JDK and JRE each supports. To generate this information for your system, type **java -version** at the command line. In the case of the Open Client toolkit, other versions, including versions from other vendors, can be used as well. When Javasoft provides a JDK and JRE for a certain platform, Progress Software includes the JDK and JRE in its distribution. For other systems, you must obtain the JDK from the system's operating system vendor. Please contact your operating system vendor for assistance in obtaining the JDK.

Table 1–1: JRE/JDK Requirements By Platform *(1 of 2)*

Platform	JDK/JRE Required	JDK Location
Sun Solaris SPARC 2.6/2.7/2.8 (32-bit)	1.3.0 (JDK and JRE Installed by Progress)	`http://www.sun.com`
Sun Solaris SPARC 8 (64-bit)	1.3.0 (JDK and JRE Installed by Progress)	`http://www.sun.com`
Sun Solaris Intel 2.6/2.7/2.8 (32-bit)	1.3.0 (JDK and JRE Installed by Progress)	`http://www.sun.com`
HP-UX 11.0 (32-bit and 64-bit)	1.3.0 (JRE Installed by Progress)	`http://www.hp.com`
Compaq Tru64 UNIX 4.0F (64-bit)/5.0 (64 bit)/5.1 (64 bit)	1.3.0	`http://www.compaq.com`
SCO 5.0.6	1.3.0	`http://www.sco.com/java`
Red Hat Linux 6.2 Intel, 7.0	1.3.0_02 SunJVM	`http://java.sun.com/j2se/1.3/download-linux.html`

Table 1–1: JRE/JDK Requirements By Platform *(2 of 2)*

Platform	JDK/JRE Required	JDK Location
RedHat Linux 7.1/7.2	1.3.1 SunJVM	`http://www.sun.com`
SuSE Linux 7.2/7.3	1.3.1 SunJVM	`http://www.sun.com`
Cladera Open Linux 3.1.1	1.3.1 SunJVM	`http://www.sun.com`

NOTE: Since Progress Version 9.1C, Java 2 incorporates the Japanese Supplement, that is the Japanese Java Development Kit (JDK) and Just in Time (JIT) updates, as part of the standard shipment. These updates allow for the support of Swing and also include Japanese localization of messages.

1.1.2 Requirements For Building Version 9 Applications

Developing a character-based Progress 4GL application requires one of the following:

- A character terminal attached to a computer
- A PC that is connected to a host computer

1.1.3 Requirements For Running Version 9 Applications

Table 1–2 lists the requirements for running Version 9 applications.

Table 1–2: Requirements For Running Version 9 Applications

Component	Requirement
Terminal	A character terminal attached to a host computer. **NOTE:** Progress Version 9 does not support spacetaking terminals unless the terminal has a firmware setup option to change it to nonspacetaking mode.
Libraries	Networking libraries must be installed on your machine. **NOTE:** Both single- and multi-user Progress work on UNIX-to-UNIX machines networking through the Progress-supported network protocol TCP/IP. You can also connect to a UNIX server from a Windows client through TCP/IP.
JDK	Installed JDK components. See Table 1–1 for the current version releases.

1.1.4 Disk Space Requirements

Table 1–3 lists the approximate file size that each Progress product requires based on a 1K-byte block size and a complete installation that includes support for one installed language. Actual space requirements can vary, depending on the block size of your machine.

Additionally, some products share common files. Therefore, an installation of several products would not necessarily require the amount of approximate disk space you calculate if you add together the Total File Size of each product, based on the information provided in Table 1–3.

NOTE: During the installation process, your system must also have extra space to contain the installation utility, installation tables, and Progress component files in compressed format. This additional space is needed only for the installation. As the installation ends, all related installation files are deleted from the current working directory.

Table 1–3: Approximate Disk Space Requirements *(1 of 2)*

Product	Total File Size
4GL Development	385MB
Application Debugger	11MB
AppServer	185MB
AppServer Internet Adapter	11MB
AppServer Internet Adapter/S	11MB
Client Access (SQL-92)	140MB
Client Connect	238MB
Client Networking	238MB
Client-Side Security	3MB
DataServer for ORACLE	211MB
Enterprise Database	262MB
ESQL 89/C	13MB
Name Server	107MB
Name Server Load Balancing	9MB
Personal Database	262MB
ProVision Development Server	368MB
ProVision Plus Development Server	412MB
Query/Results	114MB
Secure AppServer	185MB

Table 1–3: Approximate Disk Space Requirements *(2 of 2)*

Product	Total File Size
SonicMQ Adapter	107MB
Toolkit	10MB
WebSpeed Development Server	408MB
WebSpeed Messenger	14MB
WebSpeed Transaction Server	276MB
WebSpeed Transaction Server (Enterprise)	284MB
Workgroup Database	262MB

1.2 4GL Client/Database Server Compatibility

If you run a Progress multi-user system with more than one Progress version, make sure that your 4GL client and database server are compatible. Table 1–4 lists the 4GL client versions that are compatible with the Progress database server versions. All releases support TCP/IP networking.

Table 1–4: Client/Server Compatibility

4GL Client	Database Server
Version 9.1	Progress Version 8.x and Version 9.x
Version 9	Progress Versions 8.x and Version 9
WebSpeed Versions 2.0 and 2.1	Progress Version 8.x
WebSpeed Version 3.0	Progress Version 8.x and Version 9
WebSpeed Version 3.1	Progress Version 8.x and Version 9.x

1.3 Supported Platforms

Table 1–5 lists the platforms supported with this release and the minimum operating system level supported.

Table 1–5: Supported Platforms

Platform	Minimum Operating System Level
HP-UX 11 (32-bit and 64-bit)	HP-UX 11
IBM AIX (32-bit)	IBM AIX 4.3.3, AIX 5L
Sun Solaris SPARC (32-bit)	Solaris SPARC 2.6
Sun Solaris Intel (32-bit)	Solaris Intel 2.6
Compaq Tru64 UNIX (64-bit)	Compaq Tru64 UNIX 4.0F
Unixware	Unixware 7.1.1, SCO Openserver 5.0.6, Open UNIX 8
Linux	Red Hat 6.2 with kernel 2.2.16, SuSE 7.2, TurboLinux 7.0, Caldera Open Linux 3.1.1

1.4 Licensing

When installing Progress, the installation utility prompts you to enter product information from your *License Addendum*. The installation utility records the license information in the Progress configuration file, `progress.cfg`. Use the Show Configuration (SHOWCFG) Utility to display the product license information for each Progress product installed on your system:

```
install-path/bin/showcfg install-path/progress.cfg
```

NOTE: When you install a client networking license, the ADM/ADM2 directories are not installed in the `$DLC/GUI` directory. This r-code is considered part of your application and should be deployed as a module of your application.

For more information about licensing, see Chapter 3, "System Administration Utilities."

1.5 Product Life Cycle

Progress Software Corporation has developed a Product Life Cycle designed to help communicate the product development and technical support resources available during a product's life span. This section outlines the different product stages starting from the first commercial shipment (FCS) of the product to eventual product retirement, as well as the life cycle status for each Progress Software product. This information will help you develop your product plans with the knowledge and understanding of Progress Software product life cycle plans. To assist you with migration and transition strategies, Progress Software offers a wide variety of education courses and consulting services. This section does not address Progress Software's future product plans. Please be advised that this information is subject to change by Progress Software without notice. For further information, please visit our Web site at `www.progress.com or` contact your Progress Software representative.

The term "product" is used in a general sense; it refers to a combination of a particular products and versions running on a particular hardware platform and/or operating system. For example, Progress ProVision Version 9.1 for Windows refers to a particular product on a particular platform, and Progress Version 8.3 for HP-UX platforms refers to several products on HP-UX platforms.

All currently shipping products are classified in various phases of the Product Life Cycle. The Product Life Cycle definitions include specific R&D efforts, technical support services, inventory policies, and certification policies. New operating environments that will be considered for certifications include new platforms, operating systems, third-party databases, browsers, Web servers, and JVMs. Our goal is to place each product in the appropriate phase of its life cycle so that you know the level of support provided for each product. There are some situations where the placement of products in the product life cycle differs from the general policy.

Table 1–6 summarizes the Product Life Cycle phases.

Table 1–6: Product Life Cycle Phases *(1 of 2)*

Phase	New Features	New Certifications	Support	Inventory
Active	Yes	Yes	BasicService	Yes
Functionally Stable	No	Yes	BasicService	Yes

Table 1–6: Product Life Cycle Phases *(2 of 2)*

Phase	New Features	New Certifications	Support	Inventory
Mature	No	No	Limited BasicService	Yes
Retired	No	No	Limited BasicService	No, except for purchase of additional user counts for existing license.

The four phases in the Product Life Cycle are:

- **Active** — Active Phase products are fully supported and receive new features and enhancements. Progress Software recommends that customers begin all new projects with Active Phase products and migrate existing applications to Active Phase products as soon as possible. If product problems are found, Progress Software Technical Support will work with customers licensed under a BasicService maintenance contract towards an appropriate resolution. Progress Software will evaluate Active Phase products for certifications on new operating environments.

- **Functionally Stable** — Functionally Stable Phase products are fully supported but do not receive new features. If problems are found in Functionally Stable Phase products, Progress Software Technical Support will work with customers licensed under a BasicService maintenance contract towards an appropriate resolution where possible. Progress Software will evaluate Functionally Stable Phase products for certifications on new operating environments.

- **Mature** — Mature Phase products do not receive new features and support services are limited. You can continue to deploy existing applications built on Mature Phase products. However, Progress Software strongly recommends that customers use newer versions to build next-generation products. Mature Phase products will **not** be evaluated for certifications on new operating environments.

  Due to the advanced age of Mature Phase products, only Limited BasicService is available. With the knowledge and skills available, Progress Software Technical Support will provide its best efforts to resolve customer issues and answer customer questions on Mature Phase products licensed under a Limited BasicService maintenance contract. However, the knowledge and skills required to resolve issues on products supported by Limited BasicService are **not** guaranteed.

Due to the terms of Limited BasicService, Progress Software strongly recommends that customers migrate to fully supported Progress Software products as soon as possible in order to maintain the highest level of support for their applications and systems.

- **Retired** — Products placed in the Retired Phase are not available for sale except for the purchase of additional user counts for existing licenses, which do not require the delivery of media or documentation. You should not continue to develop and deploy applications with licensed Retired Phase products. Progress Software Corporation will **not** evaluate Retired Phase products for certifications on new operating environments.

  Progress Software suggests possible migration paths to similar products for Retired Phase products and platforms. Since different versions of Progress Software products have varying system requirements, it is important to consider operating system and hardware requirements when evaluating your migration options.

  With the knowledge and skills available, Progress Software Technical Support will provide its best efforts to resolve customer issues and answer customer questions on Retired Phase products licensed under a Limited BasicService maintenance contract. However, the knowledge and skills required to resolve issues on products supported by Limited BasicService are **not** guaranteed.

  Due to the terms of Limited BasicService, Progress Software strongly recommends that customers migrate to fully supported Progress Software products as soon as possible in order to maintain the highest level of support for their applications and systems.

  Progress Software provides Limited BasicService support for a specified time period after the product’s retirement date. Progress Software reserves the right to adjust the cost of Limited BasicService, which will be applied upon contract renewal. Due to the terms of Limited BasicService, Progress Software strongly recommends that customers migrate to fully supported Progress Software products as soon as possible in order to maintain the highest level of support for their applications and systems.

2

Installing Progress

This chapter contains step-by-step instructions for installing Progress products on the UNIX and Linux operating systems.

Specifically, this chapter contains the following sections:

- Performing Preinstallation Tasks
- Beginning the Installation
- Creating a Directory For a First-time Progress Version 9.1 Installation
- The JavaHome Path
- Installing Progress On Your System
- Performing a Typical Or Complete Installation
- Performing a Custom Installation
- Running the Installation Utility In Batch Mode
- Performing Postinstallation Tasks
- Downloading Executables For Heterogeneous Environments
- Installing Additional Products
- Adding Components To Previously Installed Products

NOTE: The installation time varies depending on your system, but can last up to one hour.

2.1 Performing Preinstallation Tasks

Complete the following preinstallation tasks before starting your Progress product installations:

- Read the release notes
- Install other required applications
- Upgrade an existing Progress installation (if applicable)

2.1.1 Reading the Release Notes

Your media box might contain a document entitled *Progress Release Notes*. If so, please read the hard-copy release notes before performing the installation. The *Progress Release Notes* might replace or supplement the installation instructions in this guide. In addition, online *Progress Release Notes* are available during the installation.

2.1.2 Installing Other Required Applications

If any other required applications are necessary for your platform, install them before you install Progress products.

Compaq Tru64, SCO OpenServer, UNIX, and Linux

If you are installing Progress on a Compaq Tru64, SCO OpenServer, UNIX, or Linux machine, you must first install the required version of the JRE and/or JDK (if not already installed on your system). The directory where you install the JRE and/or JDK must be in your search path when you install and subsequently run Progress. For more information on the JRE and JDK requirements, see the "System Requirements" section in Chapter 1, "Installation Requirements."

WebSpeed

For WebSpeed products, install a Web server product and configure a Web server directory on a machine where you want these products to reside. For more information on installing and configuring WebSpeed products, see the *WebSpeed Installation and Configuration Guide*.

Open Client Toolkit

The Open Client Toolkit is available for installation only on Windows. However, you can copy to a UNIX machine a portion of the toolkit, which allows you to generate Java proxies that you have initially created on Windows. To use this portion of the toolkit on UNIX, you might have to install and configure additional Java tools. For more information, see the chapter on configuration and deployment in the *Progress Client Deployment Guide*.

SQL-92

The installation program does not automatically install the JDK component when you install the Enterprise Database, the Personal Database, or the Workgroup Database with SQL Client Access. If you intend to develop Java stored procedures and Java triggers for your database, you must install a Progress development product such as ProVision. For information on writing Java stored procedures and triggers, see the *Progress SQL-92 Guide and Reference*.

2.1.3 Upgrading an Existing Progress Installation

If you have an existing Progress Version 8 or earlier installation, follow these steps before you begin to install Version 9:

1 ♦ Make sure that the ULIMIT is set to at least 8MB and at least 128 file descriptors. For specific instructions on setting the ULIMIT on your system, consult its man page by typing **man ulimit** at the command prompt.

2 ♦ Truncate the `before-image` (.bi) file of any existing database using the PROUTIL TRUNCATE BI utility. Back up your Version 8.*x* database using the PROBKUP utility. For more information on the PROUTIL TRUNCATE BI and PROBKUP utilities, see the "Administration Utilities" chapter in the *Progress Database Administration Guide and Reference*.

CAUTION: Progress Version 9 requires that your databases be converted to a multi-volume structure. If you are using single-volume databases with Progress Version 8, you must convert your Version 8 databases to a multi-volume structure before converting the databases to Version 9. (Note that you must truncate your BI file in Version 8 before converting to Version 9.) If you plan to replace Progress Version 8 files with Version 9 files, complete this step before installing Version 9 and avoid erasing your Progress Version 8 files.

3 ♦ Make copies of your `ubroker.properties` and `conmgr.properties` files to another directory. The new installation automatically upgrades the files in the *`install-path`*`/properties` directory. After you have finished your new installation, replace the newly installed versions of these files with your copies. When you start the AdminServer, your older files will be updated to match the current standards for these files.

4 ♦ Make sure you are installing the software into a directory other than the directory from which you are running the Installation Utility.

2.2 Beginning the Installation

How you install Progress depends on what products, components, and subcomponents you choose to install. You can tailor your Progress installation so that only those products and components essential to your business operation are installed. See your specific product documentation for information about which mandatory components you must install and where to install them.

For example, if you are installing the 4GL Development product and you choose the Typical installation option, the ProBuild Utility and Progress Explorer tools will not be installed because they are not mandatory components. To install either of these components you must choose either the Complete or the Custom installation option. See Appendix D, "Progress Version 9 Typical and Complete Installation Components," for a complete list of mandatory and optional components for each product.

As you proceed through the installation, you will specify one of the installation options described in Table 2–1.

Table 2–1: Installation Options

Installation Option	Components Installed
Typical	Install the mandatory components and their mandatory subcomponents of the Progress product you are installing. This default installation option is recommended for most users. If you want to perform a Typical installation, continue with the "Performing a Typical Or Complete Installation" section. If you want to see which product components and subcomponents are installed during a Typical installation, see Appendix D, "Progress Version 9 Typical and Complete Installation Components."
Complete	Install the mandatory and optional components and subcomponents of the Progress products you are installing. To perform a Complete installation, see the "Performing a Typical Or Complete Installation" section. If you want to see which product components and subcomponents are installed during a Complete installation, see Appendix D, "Progress Version 9 Typical and Complete Installation Components."
Custom	Distribute Progress components on different machines and work around space limitations. This method is recommended for advanced users. If you perform a Custom installation and do not choose any optional components or subcomponents, only those components and subcomponents that are mandatory for the products you are installing are installed (as in a Typical installation). To perform a Custom installation, see the "Performing a Custom Installation" section.

NOTE: Before you begin the Progress 9.1 installation, note the following:

- You must have root privilege on the machine on which you are installing Progress Version 9.1. For more information, see your UNIX operating system documentation.
- You can install Progress products in any order.
- If you are installing Version 9.1 and you have an existing Progress Version 8 or earlier installation, you need to perform some additional steps before you begin the Version 9.1 installation. See the "Performing Preinstallation Tasks" section.

2.3 Creating a Directory For a First-time Progress Version 9.1 Installation

If you are installing Progress 9.1 for the first time, you must create a directory where you want the Progress products to be installed. However, you should **not** create a subdirectory called DLC, because the installation utility automatically creates this for you.

2.4 The JavaHome Path

The JavaHome path is the Java installation directory. You **must** have the Java installation directory in your search path when you install and subsequently run Progress.

2.5 Installing Progress On Your System

Follow these steps to begin installing your Progress products:

1 ♦ See your *License Addendum* for additional information that you need to complete the installation. The *License Addendum* includes the version number, serial number, and control numbers for your installation. It also provides volume information, the media ID number, and the licensed user limit. Once you complete the installation process, keep the *License Addendum* in a secure area.

NOTE: When you install a client networking license, the ADM/ADM2 directories are not installed in the `$DLC/GUI` directory. This .r code is considered part of your application and should be deployed as a module of your application.

2 ♦ If you have not done so already, read the hard-copy *Progress Release Notes* in your Progress media box. The *Progress Release Notes* might replace or supplement the installation instructions in this booklet.

3 ♦ Log in as root. If you do not know the root password for your machine, check with your system administrator.

4 ♦ Insert the Progress Version 9.1 installation CD into the CD-ROM drive.

NOTE: There is an automount daemon on the Solaris platform that mounts the CD within approximately five seconds. If, however, the automount does not occur on your Solaris machine, use the mount command provided in Table 2–2.

NOTE: You **cannot** install Progress Version 9.1 on Linux over an NFS-mounted device.

5 ♦ Enter your platform-specific mount command (where *device-name* is the device you are using for the installation and *mount-point* is the mount-point directory). Table 2–2 lists the mount commands for each supported platform.

Table 2–2: Mount Commands *(1 of 2)*

Operating System	Mount Command
HP-UX 11 (32-bit and 64-bit)	`mount -F cdfs -r -o cdcase` *device-name mount-point* For example: `mount -F cdfs -r -o cdcase /dev/dsk/c0t2d0 /cdrom`
IBM AIX 4.3.3/5L (32-bit)	`mount -v cdrfs -r` *device-name mount-point* For example: `mount -v cdrfs -r /dev/cd0 /cdrom`
SCO UnixWare 7.1.1	`mount -F cdfs -r -o nmconv=lm,fperm=555,dperm=555` *device-name mount-point* For example: `mount -F cdfs -r -o nmconv=lm,fperm=555,dperm=555 /dev/cdrom/c0b0t210 /cdrom`
SCO OpenServer 5.0.6	`/etc/mount -f iso9960,lower -r` *device-name mount-point* For example: `/etc/mount -f iso9960,lower -r /dev/cdrom/c0b0t210 /cdrom`
Sun Solaris SPARC 2.6, 2.7, and 2.8 (32-bit)	`mount -F hsfs -o ro,nrr -r` *device-name mount-point* For example: `mount -F hsfs -o ro,nrr -r /dev/dsk/c0t4d0s0 /cdrom`
Sun Solaris Intel 2.6, 2.7, and 2.8	`mount -F hsfs -r -o ro,nrr` *device-name mount-point* For example: `mount -F hsfs -r -o ro,nrr /dev/dsk/c0t4d0s0 /cdrom`

Table 2–2: Mount Commands *(2 of 2)*

Operating System	Mount Command
Compaq Tru64 UNIX 4.0D	`mount -t cdfs -o noversion` *device-name mount-point* For example: `mount -t cdfs -o noversion /dev/rz3c /cdrom`
Linux	`mount -t iso9660` *device-name mount-point* For example: `mount -t iso9660 /dev/cdrom /cdrom`

6 ♦ Enter the following install command:

```
mount-point/proinst
```

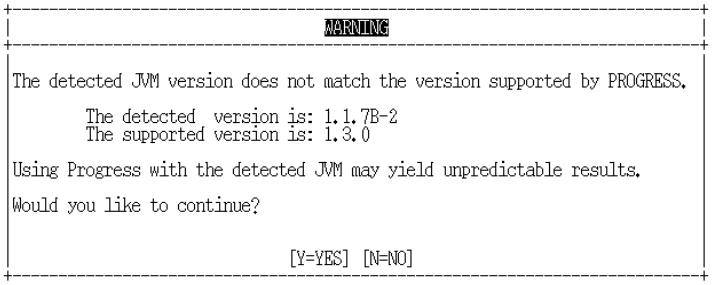
NOTE: You cannot run `proinst` if you are in the *mount-point* directory.

- If you have one of the platforms for which Progress does not ship a JVM, the Installation utility searches for a JVM on your system. (Progress ships a JVM for Solaris Sparc, Solaris Intel, and HPUX platforms.) If the Installation utility does not detect a JVM on a platform for which Progress does not ship a JVM, then the following warning message will appear:

```
+------------------------------------------------------------+
|                            WARNING                         |
+------------------------------------------------------------+
|                                                            |
|               A JVM has not been detected.                 |
|Installation of products that require JAVA will not be allowed.|
|                                                            |
|                                                            |
|                          [Enter=OK]                        |
+------------------------------------------------------------+
```

  The Installation will not continue, and you will have to install a JVM before proceeding with the Installation.

- If you have one of the platforms for which Progress does not ship a JVM, the Installation utility checks whether you have the JVM version that is supported by Progress on your system. (Progress ships a JVM for Solaris Sparc, Solaris Intel, and HPUX platforms.) If the Installation utility does not detect the supported JVM version on a platform for which Progress does not ship a JVM, then the following warning message will appear:

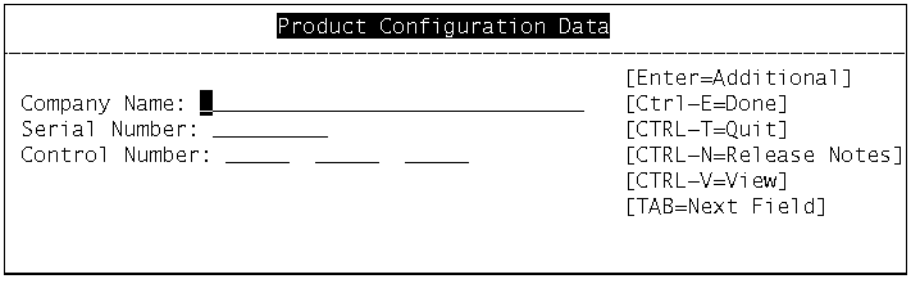


You can choose to continue with the Installation whether or not you have the supported JVM version on your system. However, Progress Software Corporation recommends you install the supported JVM version in order to ensure full functionality.

- If you have either the supported JVM version on your system or a platform for which Progress ships a JVM, then the Welcome dialog box appears:



7 ♦ Press **RETURN**. The **Product Configuration Data** dialog box appears:



8 ♦ Type your company name (any character string is acceptable), the serial number, and control numbers from your *License Addendum* for the first product you want to install. Use the **TAB** key to move to the next field.

NOTE: This information is not case sensitive. The installation process automatically converts all letters to uppercase.

Table 2–3 describes the options available on the **Product Configuration Data** dialog box.

Table 2–3: Product Configuration Data Dialog Box Options

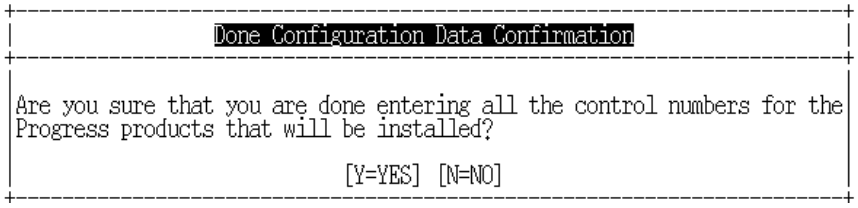
Option	Function
ENTER= Additional	Confirms that the product information entered is correct and clears the input fields for additional information for another Progress product you want to install.
CTRL-E = Done	Confirms that the product information entered is correct and continues to the next step in the installation. The actual key you use to indicate Done varies by terminal type. Refer to the Product Configuration Data dialog box for the appropriate key combination for your terminal.
CTRL-T = Quit	Quits the Installation Utility. The Quit Confirmation dialog box appears. Type **Y** to quit or **N** to continue the installation. (When you quit the installation process, the installation utility and all the installation files you copied into your working directory are deleted.)
CTRL-N = Release Notes	Displays the online *Release Notes*.
CTRL-V = View	Displays the selected Progress products for which you entered control numbers.
TAB = Next Field	Moves to the next field in the dialog box.

If you enter incorrect product configuration information, the following error message appears indicating that the data is invalid:

```
+-------------------------------------------------------------------+
|                              Error                                |
+-------------------------------------------------------------------+
|                                                                   |
|INSTALLATION ERROR (139): The entered configuration information is |
|incorrect.  Re-enter the configuration data.                       |
|                                                                   |
|                           [Enter=OK]                              |
+-------------------------------------------------------------------+
```

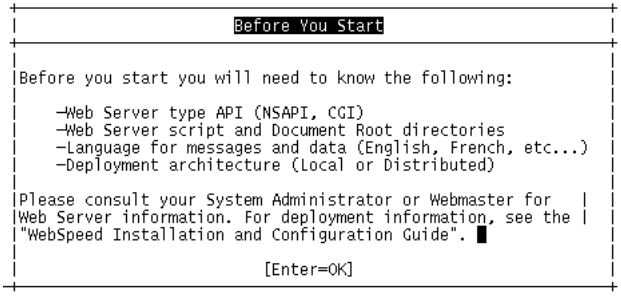
Press **RETURN** to correct the product information and continue with the installation. The Product Configuration Data dialog box reappears, allowing you to enter the serial number and control numbers for the next product to install.

9 ♦ When you finish entering the serial number and control numbers for all the products you want to install, press **CTRL-E**. The **Done Configuration Data Confirmation** dialog box appears:



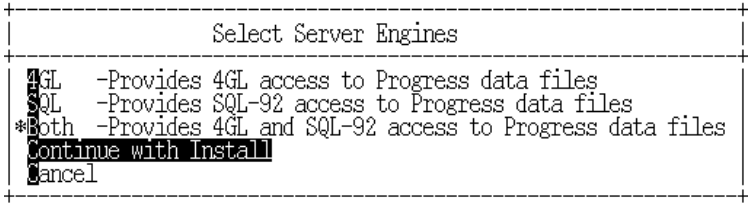
10 ♦ Type **Y** to continue with the installation. One of the following occurs:

- If you are installing a product with WebSpeed components, the **Before You Start** dialog box appears:



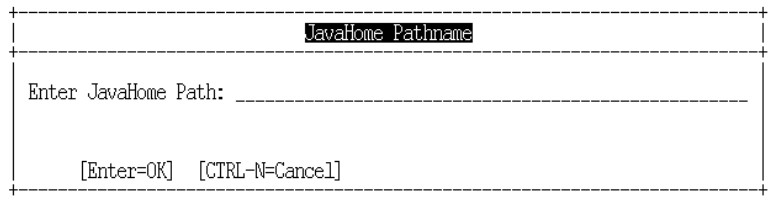
The Before You Start dialog box identifies information that you will be asked to provide later in the installation. Press **RETURN**.

- If you are installing a database product, the **Select Server Engines** dialog box appears:



Choose the **Server Engine** option. (Note that the **Both** option is selected by default.) The SQL-92 Server Engine is not available on SCO OpenServer 5 and Solaris Intel platforms. Then choose **Continue with Install**.

11 ♦ Press **ENTER** to continue. For those UNIX platforms on which the JRE/JDK is not automatically included during the installation, the following dialog box, or one that is similar, appears:



The JavaHome is the Java installation directory. The Java installation directory must be in your search path when you install and subsequently run Progress.

NOTE: On these platforms, the installation process searches for the correct version of the JDK residing on the target system before continuing the installation.

12 ♦ Enter a different version/location than the one specified, or press **ENTER** to continue. The **Type, Device and Destination** dialog box appears:

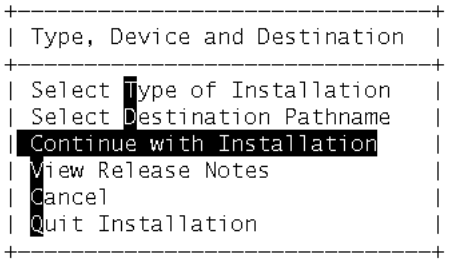


Table 2–4 describes the menu options on the **Type, Device and Destination** dialog box, which you will be working with once you choose the type of installation you want to perform.

To select an option, highlight the option using the **SPACEBAR** key, the **TAB** key, the **CURSOR** keys, or the accelerator keys that are highlighted in each selection. Press **ENTER** to execute the selected option.

Table 2–4: Type, Device and Destination Dialog Box Options

Option	Function
Select Type of Installation	Allows you to select the type of installation you want to perform: Typical, Complete, or Custom.
Select Destination Pathname	Allows you to change the default destination pathname (`/usr/dlc`) and work directory (`/usr/wrk`) pathname.
Continue with Installation	Confirms your selections (or accepts the defaults listed in the status box) and continues with the installation.
View Release Notes	Views the latest release notes using the `vi` or `emacs` editor. When you choose this option, the Installation Utility launches the editor. To exit `vi` after reading the *Release Notes*, type **:q!**. To exit `emacs`, press **CTRL-X** and then **CTRL-C**.
Cancel	Cancels the previous selection.
Quit Installation	Quits the Installation Utility. The **Quit Confirmation** dialog box appears. Type **Y** to quit or **N** to continue the installation. (When you quit the installation process, the installation utility and all the installation files you copied into your working directory are deleted.)

CAUTION: Do not make your working directory pathname a subdirectory under the destination pathname.

Progress Software Corporation recommends that you read the online *Progress Release Notes* before continuing with the installation. The *Progress Release Notes* might replace or supplement the installation instructions in this guide. The *Progress Release Notes* appear in the `vi` or `emacs` editor.

13 ♦ Continue with either the "Performing a Typical Or Complete Installation"section or the "Performing a Custom Installation"section.

2.6 Performing a Typical Or Complete Installation

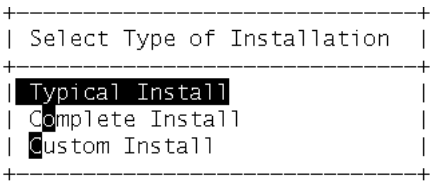
With Version 9 you can choose the installation method that best suits your company's unique product and resource requirements. You can choose one of following installation options:

- **Typical** — Installs most (but not all) of the components and subcomponents of the Progress products you install. A Typical installation saves disk space by installing only the most widely used Progress components and subcomponents. If you want to see which product components and subcomponents are installed during a Typical installation, see Appendix D, "Progress Version 9 Typical and Complete Installation Components."
- **Complete** — Installs all components and subcomponents of the Progress products you install. If you want to see which product components and subcomponents are installed during a Complete installation, see Appendix D, "Progress Version 9 Typical and Complete Installation Components."

You can also perform a Custom installation. For details, proceed to the "Performing a Custom Installation" section in this chapter.

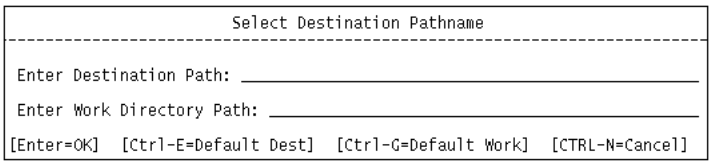
Before you proceed with the steps in this section, be sure to complete all the steps in the "Installing Progress On Your System" section. Then from the **Type, Device and Destination** dialog box, follow these steps to perform either a Typical or Complete installation:

1 ♦ Choose **Select Type of Installation** and press **RETURN**. The following dialog box appears:



2 ♦ Choose either **Typical Install** or **Complete Install**, and press **RETURN**. The **Type, Device and Destination** dialog box reappears.

3 ♦ Choose **Select Destination Pathname** and press **RETURN**. The following dialog box appears:



4 ♦ Type your Progress destination path or accept the default directory in which to install Progress (`/usr/dlc`). Then type the work directory path or accept the default provided (`/usr/wrk`).

NOTE: The actual key you use to accept the default varies by terminal type. The **Select Destination Pathname** dialog box shows the appropriate key combination for your terminal.

5 ♦ Press **RETURN**. If the directory already exists, the following dialog box appears:

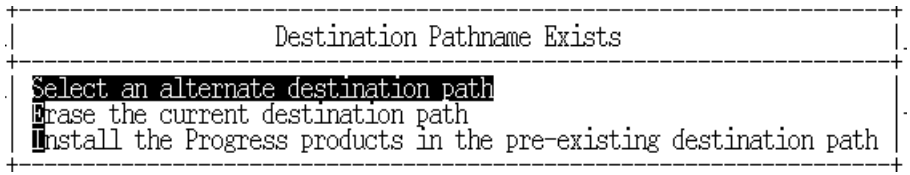


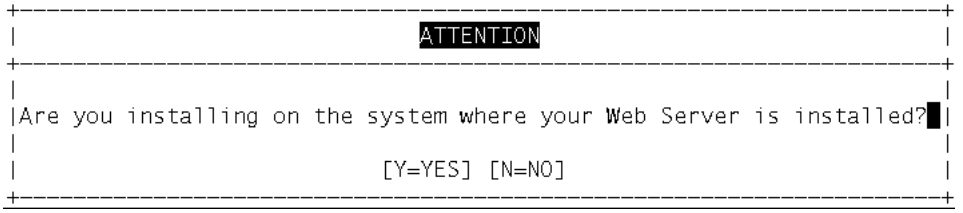
Table 2–5 lists the options in the **Destination Pathname Exists** dialog box.

Table 2–5: Destination Pathname Exists Dialog Box Options

Option	Action
Select an alternate destination path	Allows you to select another destination path. This returns you to the **Select Destination Pathname** dialog box.
Erase the current destination path	Deletes the contents and subdirectories of the named directory and reinstalls in the same directory.
Install the Progress products in the pre-existing destination path	Installs Progress products in the named directory. If the name of a file being installed matches the name of an existing file, the installation utility overwrites the existing file. The installation utility does not, however, overwrite the `progress.cfg` file.

6 ♦ Choose one of the options and make the necessary changes.

7 ♦ From the **Type, Device and Destination** dialog box, choose **Continue with Installation**. If you are installing a product with WebSpeed components, the following dialog box appears:



8 ♦ Type **Y** if you are installing Progress on the same machine where your Web server is installed. (See the *WebSpeed Installation and Configuration Guide* for information about installing WebSpeed.)

Otherwise, type **N**.

The **Language Selection** dialog box appears:

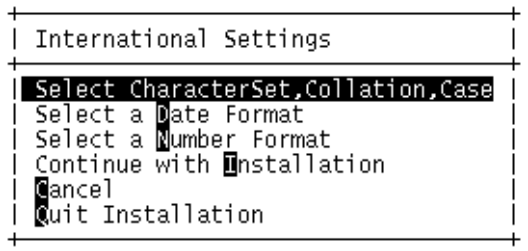


9 ♦ Select the language you want to use for error and informational messages. To select a language, use your arrow keys to highlight the language and press **RETURN**. An asterisk (*) appears next to the selected language. Select as many languages as you want.

10 ♦ You must choose one language as the default language. Press **M**, the Make Default quick key. The symbol (d) appears to the right side of the default language. If you do not select a default language, an error message will appear.

11 ♦ When you finish selecting the languages, choose **Continue with Installation**.

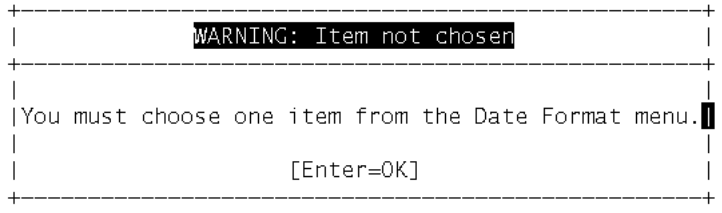
The **International Settings** dialog box appears:



The **International Settings** dialog box allows you to tailor the default Progress Startup Parameter (`startup.pf`) file to select international conventions for your locale, such as Character Set, Collation, Case, Date format, and Number format preferences. For more information on international conventions, see the *Progress Internationalization Guide*.

12 ♦ From each drop-down list, select the international Character Set, Collation, and Case; Date Format; and Number Format to tailor the default Progress `startup.pf` file.

13 ♦ Choose **Continue with Installation**. If you have not completed a required International Setting, a warning appears. For example, the following warning appears if you have not chosen the Date format:



14 ♦ Press **RETURN**, and complete the required setting.

15 ♦ Choose **Continue with Installation**, and press **RETURN**.

Once you provide all required settings, the installation performs a check of available disk space. Provided your destination path choice has enough disk space, the **Summary** dialog box appears, displaying a summary of the installation options you selected:

```
The following products will be installed:
´Client Networking´, ´WebSpeed Transaction Server´, "ORACLE DataServer´


Disk Space Required for Products: 124,917,760 bytes
Disk Space Required for Installation: 212,254,720 bytes
Disk Space Remaining After Installation: 14,564,352 bytes

Selected Destination Path: /usr1/install

Do you want to install the above listed product(s)?

                  |Y=Yes|   |N=No|
```

16 ♦ Type **Y** to continue.

The **Copy Scripts?** dialog box appears:

```
+-----------------------------+
|        Copy Scripts?        |
+-----------------------------+
|                             |
|Copy the scripts to /usr/bin?|
|                             |
|       [Y=YES] [N=NO]        |
+-----------------------------+
```

Progress products consist of several scripts and program modules. When you install a product, the scripts are placed in the installation directory you specify.

17 ♦ Choose one of the following:

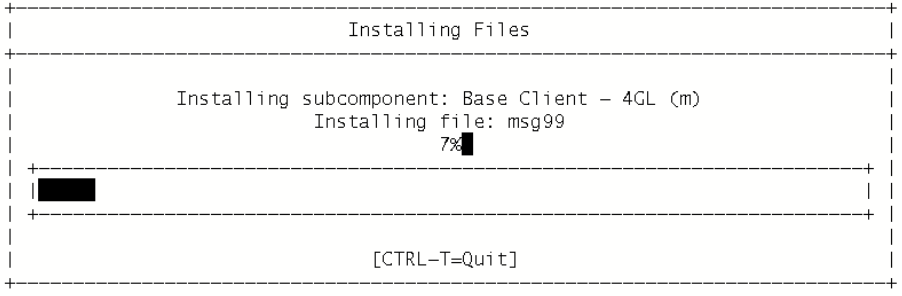
- To allow all users on your system to run the product, you should answer Yes when prompted to copy the scripts to `/usr/bin`. Type **Y** to instruct the Installation utility to place Progress 9.1 scripts in `/usr/bin` **and** the destination pathname you specified earlier.

CAUTION: Answering **Y** might cause Progress to overwrite existing scripts in this directory.

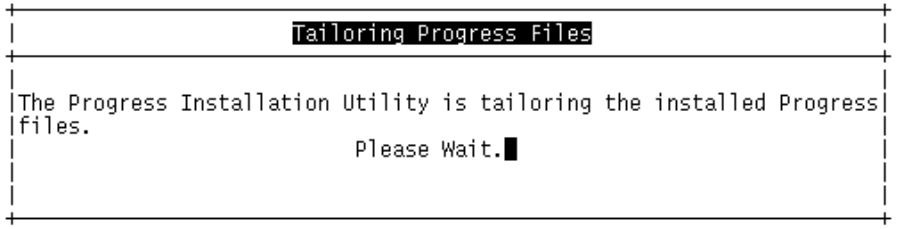
- Type **N** to instruct the Installation utility to place Progress scripts **only** in the destination pathname you specified earlier.

NOTE: If you are maintaining two versions of Progress on the same machine, answer **N** to this question.

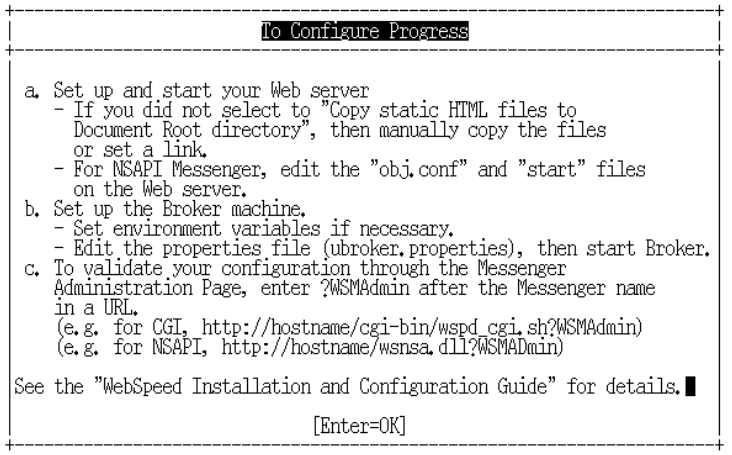
While the Progress Installation utility is decompressing the files, the following dialog box appears:



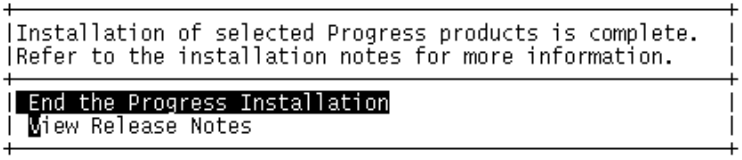
While the Progress Installation utility is tailoring the files, the following dialog box appears:



If you are installing a product with a WebSpeed component, the **To Configure Progress** dialog box appears:



18 ♦ Press **RETURN**. When the installation is complete, the following dialog box appears:



19 ♦ Choose **End the Progress Installation** to exit. Progress returns you to the UNIX system prompt.

20 ♦ You can now unmount the CD by entering the following command:

```
umount device-name mount-point
```

You are now ready to set up the user environment to run Progress Version 9.1. If you have installed an SQL product, you must set your environment variables. For more information on setting environment variables, either to support an SQL product or to customize the variables to your own preferences, see Chapter 5, "Setting Up Your Environment."

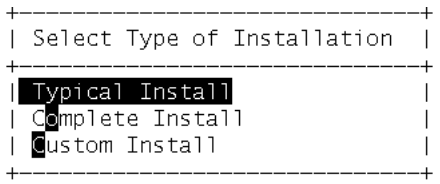
To create customized product executables, see "Building Progress Executables" in the *Progress Client Deployment Guide*. Creating executables might be required for certain product configurations.

2.7 Performing a Custom Installation

Perform a Custom installation when you want to install individual Progress product components and subcomponents. You can also choose to install only the mandatory components required for each Progress product.

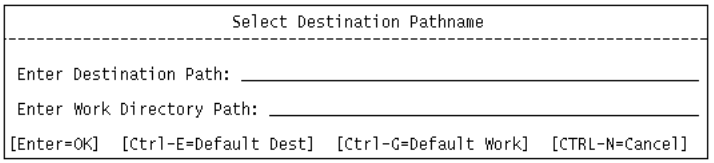
Before you proceed with the steps in this section, be sure to complete all the steps in the "Installing Progress On Your System" section. Then from the **Type, Device and Destination** dialog box, follow these steps to perform a Custom installation:

1 ♦ Choose **Select Type of Installation**, and press **RETURN**. The **Select Type of Installation** dialog box appears:



2 ♦ Choose **Custom Install** and press **RETURN**. The **Type, Device and Destination** dialog box reappears.

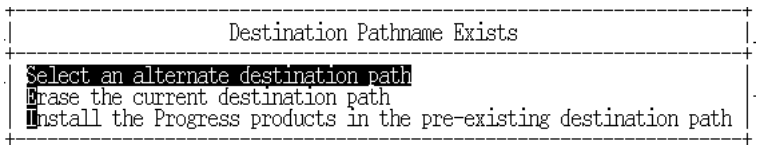
3 ♦ Choose **Select Destination Pathname** and press **RETURN**. The following dialog box appears:



4 ♦ Type your Progress destination path or accept the default directory in which to install Progress (`/usr/dlc`). Then type the work directory path or accept the default provided (`/usr/wrk`).

NOTE: The actual key you use to accept the default varies by terminal type. The **Select Destination Pathname** dialog box shows the appropriate key combination for your terminal.

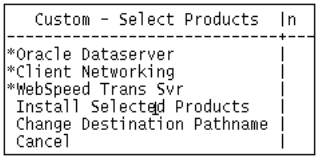
5 ♦ Press **RETURN**. If the directory already exists, the following dialog box appears:



See Table 2–5 for a list of the options in the **Destination Pathname Exists** dialog box.

6 ♦ Choose one of the options and make the necessary changes.

7 ♦ From the **Type, Device and Destination** dialog box, choose **Continue with Installation**. The **Custom - Select Products** dialog box appears:



The **Custom - Select Products** dialog box allows you to select and deselect Progress products to install. Each product you are installing is preceded by an asterisk (*).

NOTE: The symbol (m) indicates that a component or subcomponent is mandatory.

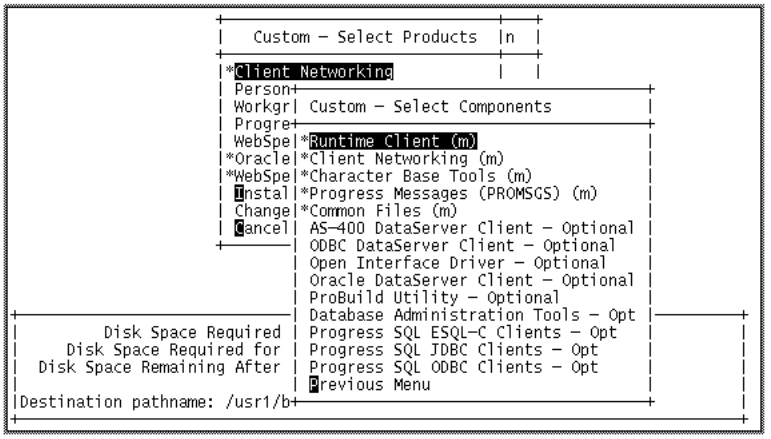
Table 2–6 describes the status box at the bottom of the **Custom - Select** Products dialog box.

Table 2–6: Custom - Select Products Dialog Box Status Information

Status Data	Description
Disk Space Required for Products	Displays the amount of disk space required to run the products you want to install.
Disk Space Required for Installation	Displays the amount of disk space required to install the products you want to install. This includes the space required to run the products plus additional space used and then released by the Installation Utility.
Disk Space Remaining After Installation	Displays the amount of disk space available after the actual installation. This includes the available disk space prior to installation, minus the disk space required for the products.
Destination pathname	Displays the destination pathname. If your destination pathname is truncated, select **Change Destination Pathname** to see the full pathname and verify its accuracy.

8 ♦ Press **RETURN** to select or deselect a product. An asterisk (*) indicates that a product is selected.

9 ♦ Press **RETURN** (again) to customize the product. The **Custom - Select Components** dialog box appears:

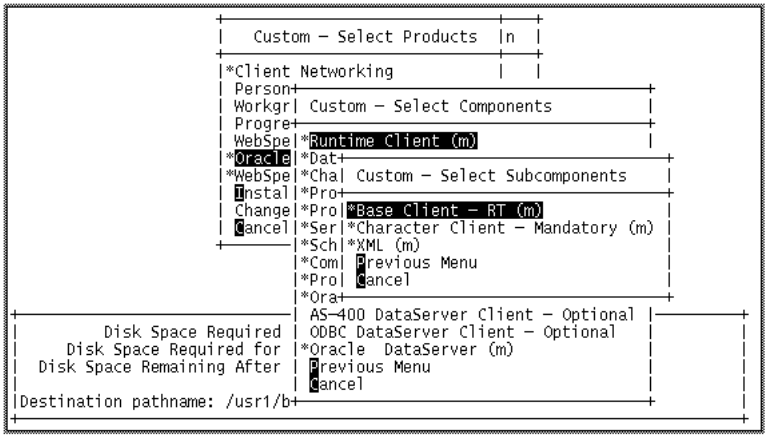


This dialog box lists the components that you can install for the selected product.

NOTE: You can choose the languages you want to install from the **Custom - Select Subcomponents** dialog box, but the **Language Selection** dialog box (described later in this section) will override your selections. Use the **Language Selection** dialog box to specify the languages you want to install.

10 ♦ Select or deselect a component to install by highlighting the component and pressing **RETURN**.

11 ♦ Press **RETURN** (again) to view the subcomponents for the selected component. The **Custom - Select Subcomponents** dialog box appears:



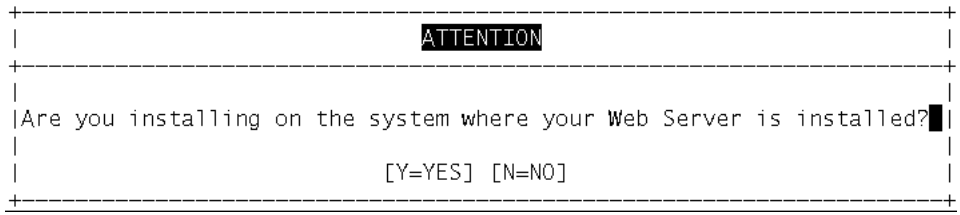
12 ♦ Select or deselect the Progress subcomponents you want to custom install for the components you selected in Step 10.

13 ♦ Choose **Previous Menu** to return to the **Custom - Select Components** dialog box.

14 ♦ Choose **Previous Menu** to return to the **Custom - Select Products** dialog box.

15 ♦ Repeat Step 8 through Step 14 for each product you are installing.

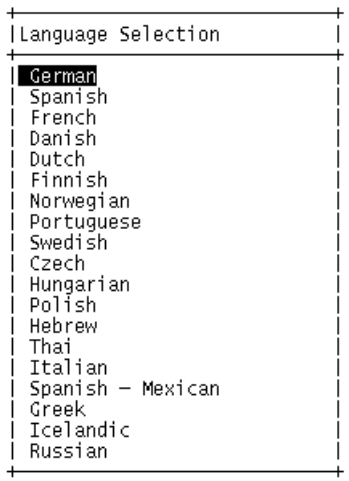
16 ♦ From the **Custom - Select Products** dialog box, choose **Install Selected Products**. If you are installing a product with a WebSpeed component, the following dialog box appears:



17 ♦ Choose one of the following:

- Type **Y** if you are installing Progress on the same machine where your Web server is installed. (See the *WebSpeed Installation and Configuration Guide* for information about installing WebSpeed.)
- Type **N** if you are installing Progress on a different machine from where your Web server is installed.

The **Language Selection** dialog box appears:

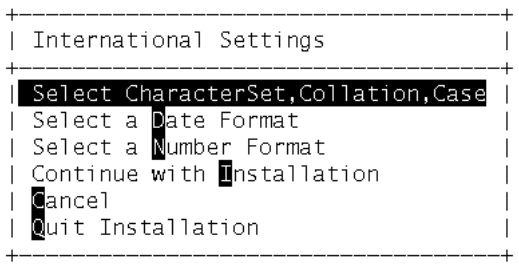


18 ♦ Select the languages you want to install. Progress can use these languages for run-time messages.

To select a language, use your arrow keys to highlight the language and press **RETURN**. An asterisk (*) appears next to the selected language. Select as many languages as you want.

19 ♦ You must choose one language as the default language. Press **M**, the Make Default quick key. The symbol (d) appears next to the default language.

20 ♦ Choose **Continue with Installation**. The **International Settings** dialog box appears:

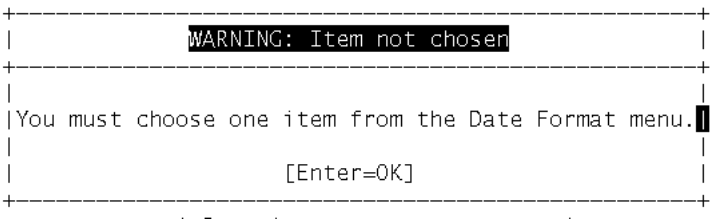


The **International Settings** dialog box allows the installation program to tailor your Progress (`startup.pf`) file to accommodate international conventions such as Character Set, Collation, Case, Date (-d) format, and Number (-numsep and -numdec) format preferences.

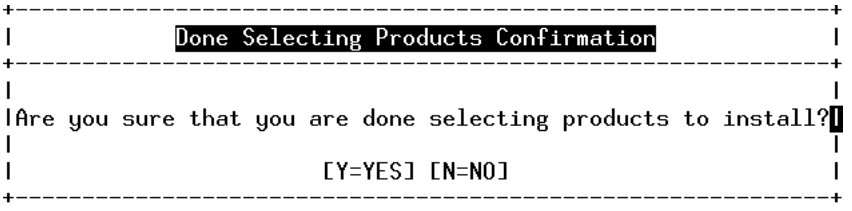
For more information on international conventions, see the *Progress Internationalization Guide*.

21 ♦ From each drop-down list, select the international Character Set, Collation, Case, Date Format, and Number Format to tailor your Progress `startup.pf` file.

22 ♦ Choose **Continue with Installation**. If you have not completed a required International Setting, a warning appears. For example, the following warning appears if you have not chosen the Date format:

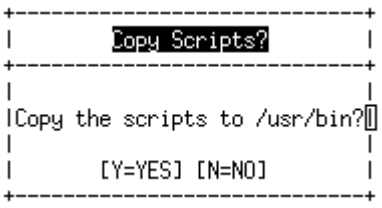


23 ♦ Press **RETURN**, and complete the required setting. The following confirmation dialog box appears:



24 ♦ Type **Y** to continue with the installation or **N** to select additional products.

The **Copy Scripts?** dialog box appears:



Progress products consist of several scripts and program modules. When you install a product, the scripts are placed in the installation directory you specify.

25 ♦ Choose one of the following:

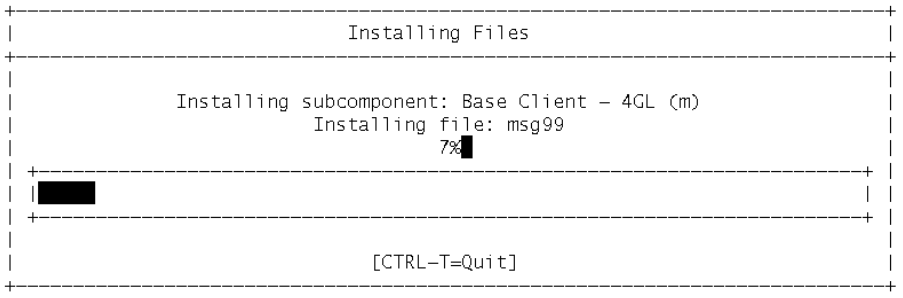
- To allow all users on your system to run the product, you should answer Yes when prompted to copy the scripts to `/usr/bin`. Type **Y** to instruct the Installation utility to place Progress scripts in `/usr/bin` **and** the destination pathname you specified earlier.

CAUTION: Answering **Y** might cause the Progress Installation utility to overwrite existing executables in this directory.

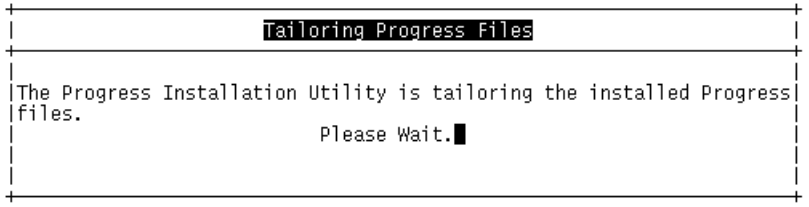
- Type **N** to instruct the Installation utility to place Progress Version 9.1 scripts only in the destination pathname you specified earlier.

NOTE: If you are maintaining two versions of Progress on the same machine, answer **N** to this question.

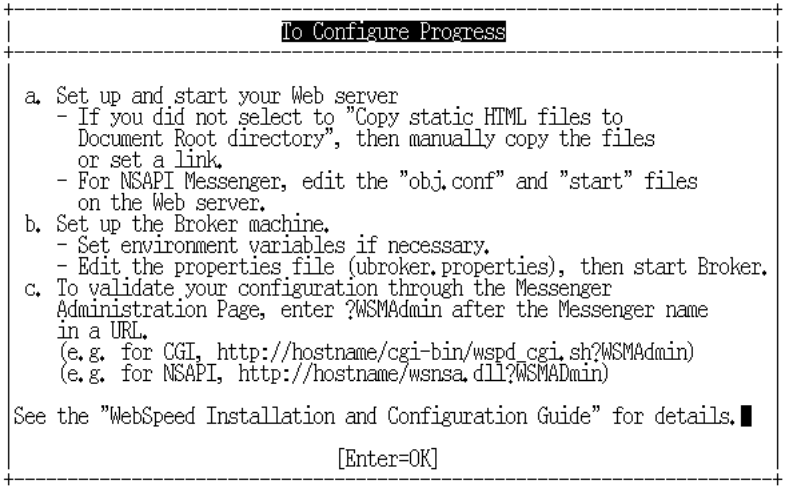
While Progress is decompressing the files, the following dialog box appears:



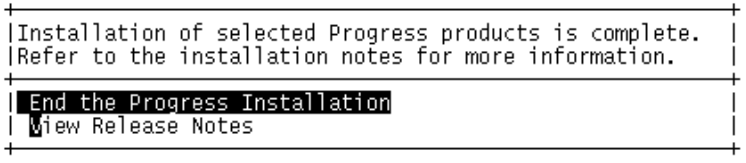
While Progress tailors the files, the following dialog box appears:



26 ♦ If you are installing a product with a WebSpeed component, the **To Configure Progress** dialog box appears:



27 ♦ Press **RETURN**. When the installation is complete, the following dialog box appears:



28 ♦ Choose **End the Progress Installation** to exit. Progress returns you to the UNIX system prompt.

29 ♦ You can now unmount the CD by entering the following command:

```
umount device-name mount-point
```

2.8 Running the Installation Utility In Batch Mode

This section describes running the Progress Installation utility in batch mode. Silent installation is only supported for Typical and Complete Installations. You cannot use this utility for a Custom Installation.

2.8.1 Support Files

Running the Progress Installation utility in batch mode requires the same support files as the interactive Progress Installation utility, plus the installation `.ini` file (*`filename`*`.ini`) containing data about the products you want to install.

2.8.2 Installation .ini File

The installation `.ini` file is user-defined. The format of the installation `.ini` file follows, where **n** represents the total number of configurations you are installing.

You can specify as many of the supported languages as you want in the Language Choice section. You can specify `lang=All` to install all of the available languages, but this is not recommended because of disk space considerations. Each language that you select will require approximately 8MB of disk space on your operating system.

The first time you install Progress you must set a default language (see the Language Default section in the example below). Be sure to type the language values exactly as they appear in the following example of an `.ini` file.

setup.ini *(1 of 2)*

```
[Configuration Count]
NumberofConfigurations=n

[Product Configuration 1]
name=Company_name
serial=serial_number
version=version_number
control=control_number_1 control_number_2 control_number_3
prodname=Product_name
.
.
.
[Product Configuration n]
name=Company_name
serial=serial_number
version=version_number
control=control_number_1 control_number_2 control_number_3
prodname=Product_name

[Language Choice]
lang=Arabic
lang=Chinese (Simple)
lang=Chinese (Traditional)
lang=Czech
lang=Danish
lang=Dutch
lang=English - American
lang=English - International
lang=Finnish
lang=French
lang=German
lang=Greek
lang=Hebrew
lang=Hungarian
lang=Icelandic
lang=Italian
lang=Japanese
lang=Korean
lang=Norwegian
lang=Persian
lang=Polish
lang=Portuguese
lang=Portuguese - Brazilian
lang=Russian
lang=Serbian
lang=Spanish
```

setup.ini *(2 of 2)*

```
lang=Spanish - Latin
lang=Spanish - Mexican
lang=Swedish
lang=Thai
lang=Turkish

[Language Default]
_szDefaultLanguage=English - American

[International Settings]
; See intlsets.txt on your CDROM for acceptable values
cpinternal=ISO8859-1
cpcollation=Basic
cpcase=Basic
dateformat=mdy
numsep=46
numdec=44

[Messenger]
InstallingWhereWebServerIsInstalled=yes


[WebServer]
; Type NSAPI
; Type CGI (If the type is CGI then the CgiScriptPath MUST exist)
WebServerType=CGI
CgiScriptPath=directory-pathname/silentCgiScriptDir
CopyHtmlFiles=yes
DocumentRootPath=directory-pathname/silentDocRootDir

[Java]
jdkHomePath=/<jdk-path>

[Server Engine]
; 4GL
; SQL
; BOTH
type=BOTH

[Type and Destination]
; Type is COMPLETE or TYPICAL
type=TYPICAL
path=directory-pathname/dlcsilent
workpath=directory-pathname/wrk
```

2.8.3 Batch Installation Using the -b Switch

The syntax for running the Progress Installation utility in batch mode follows:

```
<directory_pathname>/proinst
-b <path>/<filename.ini> -l <path>/<logfile.log>
```

In the syntax above, *directory pathname* refers to the directory where the Progress Installation utility, the installation support files, and the archive files can be found.

For example, a typical batch installation command might be:

```
proinst -b /test/install.ini -l /log/test.log
```

If no installation .ini file is specified as the argument to the switch -b, the Progress Installation utility searches the current directory for the default installation .ini file, `install.ini`. If no .ini file is found, the batch installation fails.

If no directory is specified in the argument to the -b switch, the Installation utility searches the current directory for the specified .ini file.

See the "Log File" section in this chapter for an explanation of the -l switch.

2.8.4 Log File

During a batch installation, all error messages are redirected to a log file. You can specify a log filename as an argument to the switch -l, in conjunction with the -b switch.

If no filename is specified for the log file, the Progress Installation utility uses the default log filename of `install.log`.

If no directory is specified for the log file, it is saved to the directory pointed to by the first of the following environment variables found by the Installation utility:

- TMP
- TEMP
- TMPDIR

2.9 Performing Postinstallation Tasks

Before you run Progress Version 9.1, you need to complete a few postinstallation tasks.

2.9.1 Required Tasks

The following list identifies required tasks:

- **Set Environment Variables** — For more information on setting environment variables (including SQL), see Chapter 5, "Setting Up Your Environment."
- **Create Customized Executables** — To create customized product executables, see "Building Progress Executables" in the *Progress Client Deployment Guide*. Creating executables might be required for certain product configurations.
- **Convert Existing Databases** — After your Progress Version 9 installation is complete, you must convert your Version 8 databases to Version 9 using the PROUTIL CONV89 utility. Note that if you have a Version 6 or Version 7 database, you must convert it to a Version 8 database first. For instructions on converting your Version 8 databases to Version 9, see the chapter on administration utilities in the *Progress Database Administration Guide and Reference*.

2.9.2 Setting AdminServer Security

Once you have installed Progress Version 9.1D on UNIX or Linux with products that use the AdminServer, you can optionally set user and/or group AdminServer security. You can set this option on the command line to require an individual user and/or groups of users to provide valid values during the AdminServer startup process. These products use the AdminServer: AppServer, WebSpeed, Sonic MQ Adapter, and the Progress Explorer.

In Progress Version 9.1C and earlier versions, any valid user can access and execute operations within the AdminServer. The introduction of the AdminServer user-group authorization feature allows you to require a level of security that enables only authenticated operating systems users and groups access to and use of the Admin Service.

To effectively set up this security option for your AdminServer use, review your security needs and current authenticated operating system users and groups to determine how you will set up this option during the Progress Version 9.1D installation process.

Using the Command Line In UNIX Platforms To Set the Admin Server User-Group Administration Feature

To implement the User-Group Authorization feature on a UNIX platform, you must first successfully complete the Version 9.1D installation program.

Table 2–7 identifies and briefly describes the purpose of each new command-line option:

Table 2–7: User-Group Parameter Options

Parameter name...	Syntax...	Purpose
Individual user name and password required	`-requireusername`	To require a minimum of one user ID to be resolved for each AdminServer operation before it can be executed.
Group authorization required	`-admingroup group` [{,|:}group...]	To require a minimum of one group to be resolved for each AdminServer operation before it can be executed. On a UNIX platform, a comma-separated list differentiates groups when you are specifying multiple groups on the command line.

On UNIX platforms, a group name can be any user-defined or NIS group name. UNIX can also support subgroups.

2.10 Downloading Executables For Heterogeneous Environments

The distributed architecture of Progress Version 9.1 allows you to optimize your hardware and network resources by installing components across networked machines, specifically when you are installing the WebSpeed Transaction Server, the AppServer, or the Open AppServer. Although some of these products' components must reside together on the same machine, you can, in some cases, distribute components to different machines, even if the machines run on different platforms. For example, you can install a WebSpeed Messenger or the NameServer on a UNIX platform and install a WebSpeed Broker and agents on Windows NT.

If you need either the WebSpeed Messenger executable or the NameServer executable for a platform other than UNIX, you can download the executables free of charge from `http://techweb.progress.com/services/support/ts-dnlod.htm`.

2.11 Installing Additional Products

To add other products to your current installation, follow the steps in the "Beginning the Installation" section.

NOTE: When you add products to an existing installation, you can use the installation utility in batch mode as long as you are performing a Typical or Complete installation of the products you are adding.

When the **Welcome** dialog box appears, follow these steps:

1 ♦ Press **RETURN** to continue. The **Serial & Control Numbers** dialog box appears.

2 ♦ Enter only the control numbers for the products you are adding to the list of previously installed products.

3 ♦ When you are done, press **CTRL-E**. The **Done Configuration Data Confirmation** dialog box appears.

4 ♦ Press **Y** to continue (or press **N** to add more products). The **Type Device and Destination** dialog box appears.

5 ♦ Choose **Select the Destination Pathname**, and type the path of the initial installation.

6 ♦ Press **RETURN**. The **Destination Pathname Exists** dialog box appears:

```
+-------------------------------------------------------------------+
|                   Destination Pathname Exists                     |
+-------------------------------------------------------------------+
| Select an alternate destination path                              |
| Erase the current destination path                                |
| Install the Progress products in the pre-existing destination path |
+-------------------------------------------------------------------+
```

7 ♦ Choose **Install the Progress products in the pre-existing destination path** and press **RETURN** to continue with the installation.

If you install products that affect your previously installed products, you might see the following caution message:

```
                  Release Compatibility Caution
-------------------------------------------------------------------

CAUTION: The Installation Utility has detected that you will modify a
previous installation.  Check the installation notes for release
compatibility.

Do you want to continue?

                            [Y=YES] [N=NO]
```

8 ♦ Choose **Yes** to continue with the installation.

The installation program adds your Progress products to your Version 9.1 directories automatically.

2.11.1 Changes In the Progress Core Installation Program For Progress Messages Files

Progress Version 9.1D incorporates specific file protection measures to accommodate files associated with Progress add-on products. For purposes of this document, the term add-on products refers to Progress products released independently of a point or major Progress product release. Add-on products provide functionality that enhances and is adjunct to the Progress Core software product set.

In Progress Version 9.1D, these file protection and detection procedures within Progress Core have been formalized to ensure that you have the most recent `promsgs` files. All Progress products use one centralized method to display Progress messages contained in the `promsgs` file. With each Progress add-on product you install to the Version 9.1D core installation environment, an updated `promsgs` file is installed into the Progress Core product's destination directory. Add-on installation processes have been modified such that if the add-on product contains a newer `promsgs` file than the associated core release, the following activities occur:

- The add-on product's `promsgs` file is compared with the core product's `promsgs` file to determine which of the two files is newer.
- The newer file is copied to the Version 9.1D core directory.

To support this process, Progress Version 9.1D also introduces procedures to ensure that message files are not incorrectly overwritten, and files such as `messages.jar` and Prohelp\`msgdata` that are associated with `promsgs` files are also protected.

The information in this section describes

- Details About the Installation and Update of Promsgs Files
- Procedures to Protect Files from Being Overwritten
- Procedures to Ensure Promsgs Files are Synchronized

Details About the Installation and Update Of Promsgs Files

The Progress Core Virtual Machine (PCVM) displays Progress-specific messages that are contained in promsgs files. During the Progress Core Installation process, the user selects the language(s) that can be used during the product's execution. It is possible to have several translated promsgs files installed into the progress destination path\prolang subdirectory due to this selection process. During the Core installation process, the promsgs files for the language identified as the default language are copied from the progress destination path\prolang subdirectory to the DLC directory.

Promsgs files contain the most up-to-date messages at the time the Progress core installation is released. However, the promsgs files are constantly being updated. Consequently, add-on products and Progress install service packs that are released after the Progress Core product release date can contain even more recently updated promsgs files. As each Progress add-on product is installed, the installation program checks to ensure that the newest copy of the promsgs file is being used by all products; all products use the centrally-located copy of the promsgs file stored in the DLC directory.

Procedures To Protect Promsgs Files From Being Overwritten

During the Progress Core Installation process prior to Progress Version 9.1D, certain functions have the potential to incorrectly overwrite or update a promsgs file in the local directory. To eliminate the possibility of these activities occurring, Progress Version 9.1D introduces two changes to protect promsgs files and any associated files, and to ensure that you always have the most recent promsgs files:

- A file protection mechanism has been added to the Version 9.1D Progress Core Product installation program which prohibits overwriting any Progress-associated promsgs file that already exists.

  In Progress Version 9.1D, the Progress Core Installation program default behavior determines that if a promsgs file exists in the local directory, it is the latest version. Therefore, there is no need to perform any file changes.

- The Progress Core Installation program supports a versioning scheme that adds date information to the header of the promsgs file. The install program uses this date information to help determine the latest version of a promsgs file.

If the user installs an add-on product using the Progress Add-on Installation program, this program automatically reads and compares the header information of the source promsgs file (specifically, the file associated with the add-on product) with the header information of the destination promsgs file (the file already installed in the Progress Core Installation program). Based on details noted in the header, the Add-on Installation program determines whether or not to install the promsgs file. These details include a date stamp, the single criterion for determining the most up-to-date promsgs file. The promsgs file with the latest date is always selected to be installed.

NOTE: Header information provides important information to keep promsgs files synchronized. For information about file synchronization, see the section "Procedures To Ensure Promsgs Files Are Synchronized" later in this section.

These measures also prohibit the following files that are associated with promsgs files from being overwritten by the Progress Core Installation program:

- Messages.jar
- Promsgs
- Prohelp/msgdata

Procedures To Ensure Promsgs Files Are Synchronized

Beginning in Progress Version 9.1D, promsgs files are considered to be either in-synchronization or out-of-synchronization. These terms reflect the status of the date stamp associated with a promsgs file when the date in the header of the promsgs files located in the add-on directory is compared with the date in the header of the promsgs files currently installed in the Core installation's DLC directory.

In Progress Version 9.1D, the Progress Core Installation and the Add-on Installation processes are designed to compare and evaluate the date stamp information. A promsgs file is considered synchronized if, at the conclusion of any product installation process, the Progress Core installation contains the promsgs file with the most current, or latest, date stamp. A promsgs file is considered out-of-synchronization, and therefore invalid, when the date stamp associated with the promsgs file does not display the most current date.

Table 2–8 identifies the general installation sequence that can occur at a customer site when Progress Core products and add-on products are installed. It illustrates how the promsgs files are compared, evaluated, and updated to ensure that the promsgs files are always synchronized.

Table 2–8: Promsgs File Synchronization Process *(1 of 2)*

Install Sequence	When. . .	Then. . .
1	Progress Version 9.1D core products are initially installed.	The promsgs files associated with the languages selected by the user during the install process are installed to the DLC directory.
2	A Progress add-on product is intalled.	Date stamp information in the header of the existing promsgs file in the DLC directory is compared with the date stamp information in the header of the add-on product's promsgs file. If the core promsgs file's date is *later* than the add-on product's promsgs file's date, the file is already synchronized and no changes occur. If the core promsgs file's date is *earlier* than the add-on product's promsgs file's date, the add-on promsgs file replaces the existing promsgs file.

Table 2–8: Promsgs File Synchronization Process *(2 of 2)*

Install Sequence	When. . .	Then. . .
3	Progress Version 9.1D core products are re-installed to add a new product and a new promsgs file.	Two comparisons and their associated activities occur: If the re-installation process finds that a promsgs file exists, the exisisting promsgs file is not overwritten. If, during the re-installation process, a new language is added, the promsgs file associated with that new language is installed into the DLC directory.
4	Another Progress add-on product is installed.	The date stamp information in the header of the exisiting promsgs file in the DLC directory is compared with the date stamp information in the header of the add-on product's promsgs file. If the core promsgs file's date is *later* than the add-on product's promsgs file's date, the file is already synchronized and no changes occur. If the core promsgs file's date is *earlier* than the add-on product's promsgs file's date, the add-on promsgs file replaces the existing promsgs file.

Table 2–9 illustrates another example of how this process works, using more detailed data for you to review.

The first column of Table 2–9 elaborates on the typical installation sequence outlined earlier in this section. In Step 1, the user initially installs ProVision Plus with a promsgs file for American English. The file header date of this newly installed promsgs file is 04/14/2002. In Step 2, when the user installs an add-on product, the Add-on product installation compares the header date of its American English promsgs file, 04/15/2002, with the header date of the existing American English promsgs file, 04/14/2002. Since the header date of the promsgs file associated with the add-on product is later than the existing promsgs file, the promsgs file is updated, or synchronized.

This example helps to illustrate the criterion for updating promsgs files; Only promsgs files associated with languages that are currently installed in the Progress Core will be updated by the Add-on installation process.

In Step 3, when the user installs another Progress Core product, the Progress AppServer, and identifies the Spanish `promsgs` file, the `promsgs` file with the date of 04/14/2002 is installed. This latter part of the example illustrates how the `promsgs` files can become out of sync per the date information in the respective headers.

Table 2–9: Example Of Promsgs Files Being Out Of Sync

Installation step order	Install...	And the promsgs file is...	Which contains this header date...
1.	A core product such as ProVision Plus, Version 9.1D	Installed for American English	04/14/2002
2.	An add-on product.	Updated for American English	04/15/2002
3.	A core product such as AppServer, Version 9.1	Installed for Spanish `promsgs`	04/14/2002

As Table 2–9 indicates, the installation of previously non-existing Spanish `promsgs` file dated 04/14/2002 into the Progress core installation is now out of synchronization with the updated American English `promsgs` file dated 04/15/2002, that updated the original American English `promsgs` file.

Addressing Synchronization Issues

When an additional Progress installation is performed and the Progress Core Installation program detects that a promsgs language has been installed that did not previously exist as illustrated by Step 3 in Table 2–9, the Progress Core Installation program displays a message. This message indicates the following information:

- The add-on product name that contains the latest promsgs
- Indicates the destination path of the add-on product

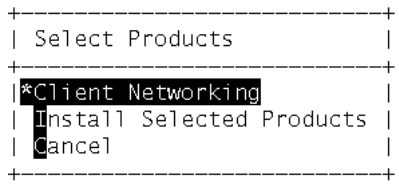
The Progress Core installation message only displays this message when it detects that add-on products have been installed and it reads a new file called `addons`. The `addons` file is a text file defined as a Windows initialization (.ini) file. This file is created and/or updated in the Progress core destination directory by the Add-on installation program. To resynchronize your `promsgs` file, you must re-install your add-on product.

2.12 Adding Components To Previously Installed Products

This new feature allows you to add components and subcomponents to existing Progress Version 9.1 and later installations without having to enter any data other than the required components or subcomponents. In earlier Progress versions it is necessary to re-install Progress and execute the “Custom Install” setup type; these steps are removed with this new Add feature.

Follow these steps to add components or subcomponents on a UNIX platform using the new add feature:

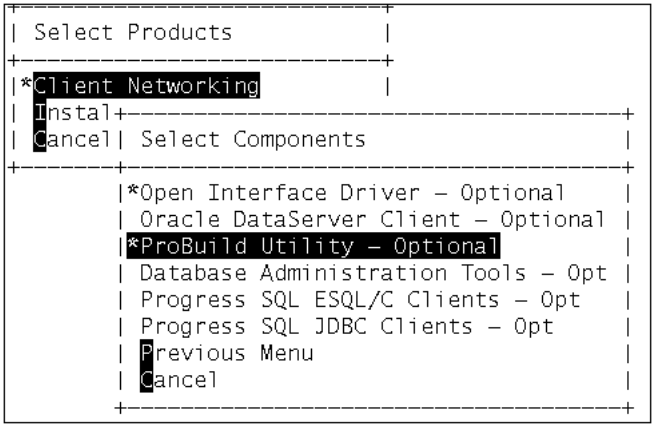
1 ♦ At the command line, type the shell script **destination-path/proaddcomp** to run the add feature. The **Select Products** dialog box appears:



All previously installed products appear on this list. The **Select Products** dialog box allows you to select and deselect Progress products for which you want to add components or subcomponents.

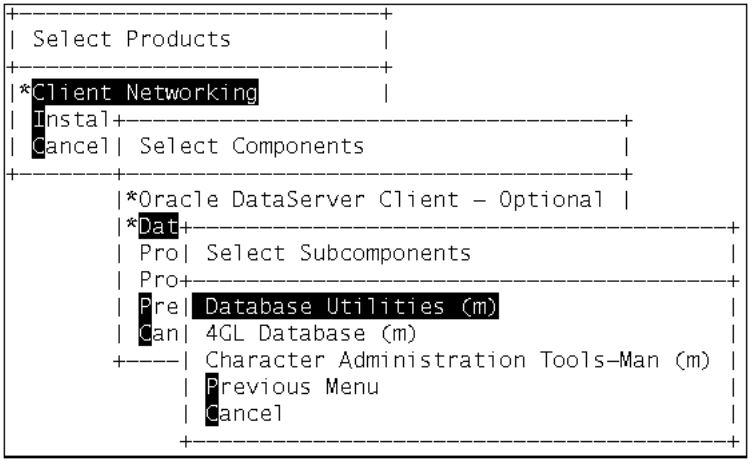
2 ♦ Select or deselect a product by highlighting the product and pressing **RETURN**. An asterisk (*) indicates that a product is selected. If you want to select the first product on the list,

you must first press **RETURN** to deselect the product and then press **RETURN** again to select it. When you select a product the **Select Components** dialog box appears:



The **Select Components** dialog box lists only those components that have not been previously installed. Select or deselect a component to install by highlighting the component and pressing **RETURN**. An asterisk (*) indicates that a component is selected.

3 ♦ If the selected component has subcomponents the **Select Subcomponents** dialog box appears:



The **Select Subcomponents** dialog box lists the subcomponents for the component you selected. The symbol (m) indicates that a subcomponent is mandatory.

Select or deselect a subcomponent by highlighting the component and pressing **RETURN**. An asterisk (*) indicates that a subcomponent is selected.

Choose **Previous Menu** and press **RETURN** when you have selected all the subcomponents you want to add.

NOTE: If you do not select all the subcomponents marked as mandatory, the **Mandatory Subcomponents** dialog box appears:

```
+-------------------------------------------------------------------------------+
|                          Mandatory Subcomponents                              |
+-------------------------------------------------------------------------------+
|                                                                               |
|Subcomponent -- to --> Component relation                                      |
|=======================================                                        |
|'4GL Database (m) -> Database Administration Tools - Opt'                      |
|'Character Administration Tools-Man (m) -> Database Administration Tools - |
|Opt'                                                                           |
|                                                                               |
|CAUTION: If you DO NOT install the above listed subcomponents the              |
|associated component may not work properly.                                    |
|                                                                               |
|Do you want to install the mandatory subcomponents?                            |
|                                                                               |
|                              [Y=YES] [N=NO]                                   |
+-------------------------------------------------------------------------------+
```

Type **Y** to return to the **Select Subcomponents** dialog box, or **N** to disregard this warning.

If you answer No to the Mandatory Subcomponents warning, the **Mandatory Confirmation** dialog box appears:

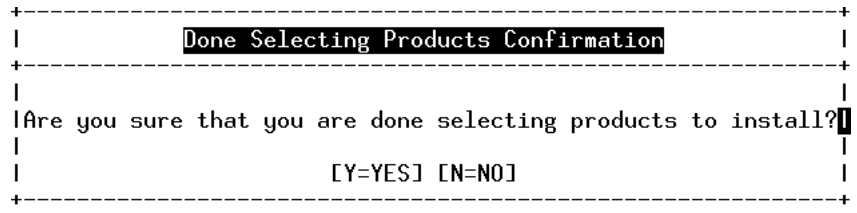
```
                  +-------------------------+
                  | Select Products         |
                  +-------------------------+
                  |*Client Networking       |
                  | Install+-------------------------------------+
                  | Cancel| Select Components                   |
+-----------------------------------------------------------------+
|                        Mandatory Confirmation                   |
+-----------------------------------------------------------------++
|                                                                 ||
|Are you sure you want to continue even though the products may not work|+
|properly?                                                        ||
|                                                                 ||
|                           [Y=YES] [N=NO]                        ||
+-----------------------------------------------------------------+|
                              | Cancel                             |
                              +------------------------------------+
```

Type **Y** to disregard this warning, or **N** to return to the **Select Subcomponents** dialog box.

NOTE: The products you install might not work properly if you do not install all mandatory subcomponents.

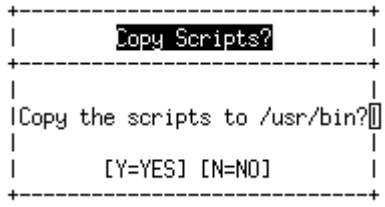
4 ♦ When you have selected all the components and subcomponents you want to add, choose **Previous Menu** and press **RETURN** to return to the **Select Products** dialog box.

5 ♦ Choose **Install Selected Products** from the **Select Products** dialog box and press **RETURN**. The **Done Selecting Products** dialog box appears:



6 ♦ Type **Y** to continue with the installation or **N** to select additional components or subcomponents.

The **Copy Scripts?** dialog box appears:



Progress products consist of several scripts and program modules. When you install a product, the scripts are placed in the installation directory you specify.

7 ♦ Choose one of the following:

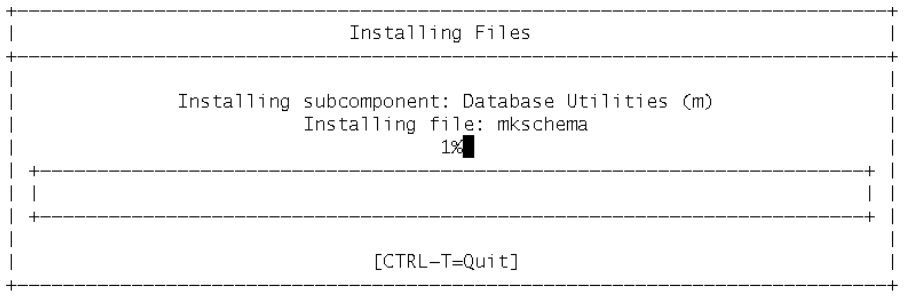
- To allow all users on your system to run the product, you should answer Yes when prompted to copy the scripts to `/usr/bin`. Type **Y** to instruct the Installation utility to place Progress scripts in `/usr/bin` **and** the destination pathname you specified earlier.

CAUTION: Answering **Y** might cause the Progress Installation utility to overwrite existing executables in this directory.

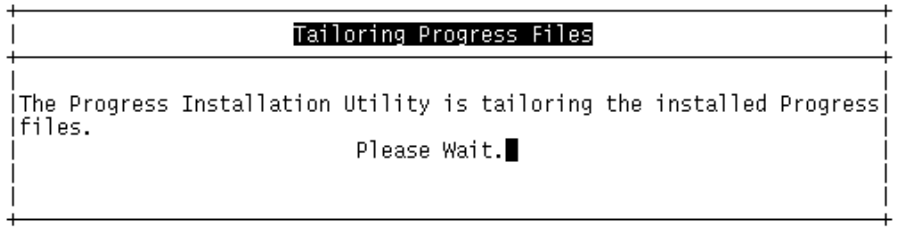
- Type **N** to instruct the Installation utility to place Progress Version 9.1 scripts only in the destination pathname you specified earlier.

NOTE: If you are maintaining two versions of Progress on the same machine, answer **N** to this question.

While Progress decompresses the files, the **Installing Files** dialog box appears:



While Progress tailors the files, the **Tailoring Progress Files** dialog box appears:



8 ♦ Press **RETURN**. When the installation is complete, Progress returns you to the UNIX system prompt.

3

System Administration Utilities

This chapter contains the following sections:

- Using the Product Update Utility
- Using the SHOWCFG Utility
- Managing User Licenses
- Progress Event Logging

3.1 Using the Product Update Utility

Use the Product Update utility, `proupdt`, to change the number of users or the expiration date for a Progress product. Contact your Progress Software Sales Representative for a new *License Addendum* if you need to use this utility.

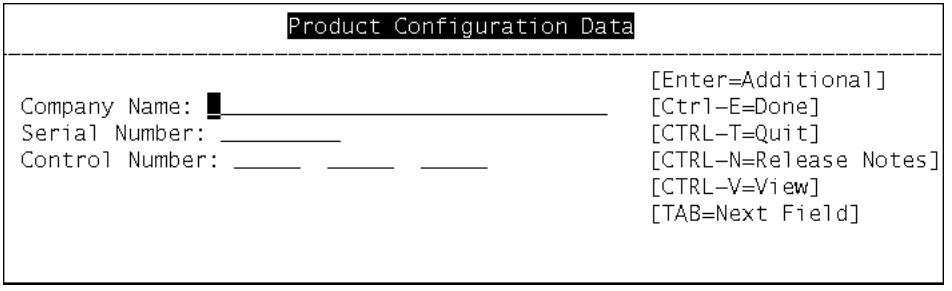
NOTE: You cannot use the Product Update utility to add additional Progress products. See the "Installing Additional Products" section in Chapter 2, "Installing Progress," for information about adding more Progress products.

Follow these steps to use the Product Update utility:

1 ♦ Change your current working directory to the directory where you installed Progress.

2 ♦ At the system prompt, type **proupdt** and press **RETURN**.

3 ♦ When a welcome message appears, press **RETURN**. The **Product Configuration Data** dialog box appears:



4 ♦ Type your company name and the new serial number and control numbers Progress Software Corporation supplies when you upgrade your license.

5 ♦ Press **ENTER**. The **Modified Product Information** dialog box appears:

```
  Modified Product Information


Product Name: Client Networking

      Old User Count:     5
      New User Count:    10
 Old Expiration Date: 12/23/1998
 New Expiration Date: 12/23/1999


           [Enter=OK]
```

Press **ENTER** again to return to the **Product Configuration Data** dialog box.

6 ♦ Press **CTRL-E** to indicate that you are done.

NOTE: You cannot press **CTRL-E** from the **Serial Number** field.

3.2 Using the SHOWCFG Utility

The Progress installation program prompts you to enter product information contained in the *License Addendum*, which is included in your software package. The installation program records license information in the Progress configuration file `progress.cfg`. You can use the Show Configuration (SHOWCFG) utility to display product installation and configuration information for each Progress product installed on your system.

The SHOWCFG utility has the following syntax:

SYNTAX

```
showcfg $env_variable/progress.cfg
```

For example:

```
showcfg $DLC/progress.cfg
```

The SHOWCFG utility displays the product configuration information stored in the Progress Version 9.1 configuration file `progress.cfg`, which is created and modified during product installation.

Figure 3–1 shows a typical display of the product configuration.

```
Configuration File:  progress.cfg

      Company Name:  Progress

      Product Name:  4GL Development
 Installation Date:  Thu Apr  5 15:35:48 2001
   Expiration Date:  12/31/2002
     Serial Number:  123456789
   Control Numbers:  9VYG9 - 7AQ7N - BVG7J
    Version Number:  9.1C
      Machine Class:  EB
        Port Number:  11
```

Figure 3–1: Product Configuration Displayed Using SHOWCFG Utility

DISPLAY FIELDS

The display contains the following information for each Progress product installed on the system:

Product Name — Name of the installed product.

Installation Date — Date the product was installed.

Expiration Date — Date the license expires.

Serial Number — Number associated with the license agreement.

Control Numbers — Numbers used by the Progress installation software.

Version Number — Software product version.

Machine Class — Tier code associated with the license agreement.

Port Number — Platform on which the software product is installed.

3.3 Managing User Licenses

The Progress user license you purchase determines how many concurrent users are allowed to run your Progress products. You are responsible for making sure users comply with your license agreement. Progress products provide reporting capabilities to help you ensure compliance with your license agreement.

The following sections describe:

- The Progress license information that is shipped with your Progress product
- How to read the Progress usage log file
- How to produce a report of current licensed user connections

3.3.1 Progress License Information

The *License Addendum* that is shipped with your Progress product provides specific information about the product license you purchased, including:

- A serial number
- A control number
- The maximum number of concurrent Progress users allowed by the license

When you install Progress Version 9.1, the installation procedure prompts you to enter product information from the *License Addendum.* The installation procedure records the license information in the Progress Version 9.1 configuration file `progress.cfg`. Use the SHOWCFG utility to display the product license information for each Progress product installed on your system.

For more information on the SHOWCFG utility, see the "Using the SHOWCFG Utility" section in this chapter.

3.3.2 Using the Progress License File

Progress creates a license file that records license-related information about Progress Version 9.1 database users. If the license file does not already exist, the broker creates it and places it in the same directory as the database (.db) file. The broker creates the file in the format *`databasename`*`.lic`, where *`databasename`* is the name of the database where the user connects.

NOTE: If Progress encounters an error while trying to open or write to the license file, the error is recorded in the database .lg file and no more entries are written to the license (.lic) file.

Reading the License File

Use a text editor to display the license file contents. The contents appear in the following order:

1. Current date
2. Current time
3. Number of licensed users specified by the configuration file
4. Current number of total connections
5. Maximum number of total connections
6. Minimum number of total connections
7. Current number of interactive connections
8. Maximum number of interactive connections for the past hour
9. Minimum number of interactive connections for the past hour
10. Current number of batch connections
11. Maximum number of batch connections for the past hour
12. Minimum number of batch connections for the past hour

For example, the following sample file entry illustrates the log format:

```
4/24/01 9:00 25 18 23 11 17 20 11 1 5 0
```

When Progress writes to the license file, the maximum and minimum values are reset for the next hour.

Maintaining the License File

The database or system administrator should consider archiving license files periodically. In one year, a license file accumulates 8,760 entries. These entries occupy about 440,000 bytes of disk space.

Since the license file must be closed before the administrator archives it, the administrator must first shut down the database. At that point, the license file can be either archived immediately or renamed and archived later.

Creating a Usage Report

To produce a report of license-related information about current Progress Version 9.1 database users, run the `licrpt.p` procedure file. The report generator input screen appears:

```
Enter Date Range:      To:        Enter Start Time (hours 0 to 23):
Enter Stop Time (hours 0 to 24):
Enter time division (in hours, or 0 for complete range):
Database Name:
```

This is a sample output from the `licrpt.p` procedure file:

```
                          Database Connection Counts
-------------------------------------------------------------------------------
| Date   Period  LcnUsers MaxTot Excptns MinTot AveTot MaxBat MinBat AvBat
|
|------- ------- -------- ------ ------- ------ ------ ------ ------ -----
|
| 04/20/01 8-17      100     20       0      0    10.      0      0     0.
|
| 04/23/01 8-17      100     20       0     18    19.      0      0     0.
|
| 04/24/01 8-17      100     23       0     17    20.      0      0     0.
|
| 04/25/01 8-17      100     33       0     17    25.      0      0     0.
|
| 04/27/01 8-17      100     32       0     26    29.      0      0     0.
|
| 04/30/01 8-17      100     26       0     17    22.      0      0     0.
```

3.4 Progress Event Logging

Progress logs significant database events such as Progress startup parameter settings; startup, shutdown, and system error messages; and application-related events. This section discusses:

- Types of messages written to the Progress event log
- Logging in the Progress event log file

3.4.1 Progress Event Log File

The Progress event log is a text file that contains a history of significant database events, such as Progress startup parameter settings and startup, shutdown, and system error messages. This file has a .lg extension.

3.4.2 Managing the Progress Event Log File Size

The event log (LG) file expands as you use the database. If it becomes too large, you can reduce its size by removing old log entries. To remove log entries from an LG file, use the Progress Log Maintenance (PROLOG) utility or a text editor. Do not remove entries from the log file while the database is in use.

Enter the following command to remove entries from an event log file:

```
prolog database-name
```

The PROLOG utility removes all but the most recent entries from the log file. For more details, see the description of the PROLOG utility in the *Progress Database Administration Guide and Reference*.

4

Troubleshooting

This chapter provides issues to consider when troubleshooting an installation.

Specifically, the chapter contains the following sections:

- Error Messages
- Altered Or Missing progress.cfg File
- Tailoring Startup Scripts

4.1 Error Messages

Table 4–1 lists some error messages, their probable causes, and where to find their solutions.

Table 4–1: Error Messages

Error Message	Cause	Solution
Unable to read `progress.cfg`, reason -1. (1732)	The `progress.cfg` file is altered or missing.	See the"Altered Or Missing progress.cfg File" section in this chapter.
Module-name not found.	The environment variables are not set correctly or not installed.	See the "Tailoring Startup Scripts" section in this chapter.
Error 304 and 305.	The ulimit is set too low.	Reset your ulimit.

4.2 Altered Or Missing progress.cfg File

If you receive the following error message, the `progress.cfg` file has been altered or deleted from the directory where you installed your Progress Version 9.1 products:

```
Unable to read progress.cfg, reason=-1.
```

Reinstall the Progress Version 9.1 product.

CAUTION: Do not alter or delete the `progress.cfg` file, or the Progress broker startup will fail.

Table 4–2 lists the reasons for an altered or missing `progress.cfg` file.

Table 4–2: Reasons For Altered Or Missing progress.cfg File

Reason	Description
-1	Could not find `$DLC/progress.cfg`.
-4	Bad checksum. Invalid file.
-6	Could not read the specified number of bytes; the file is truncated.
-7	Could not allocate enough memory to read the configuration file.

4.3 Tailoring Startup Scripts

Typically, the installation procedure automatically tailors the startup scripts for the Progress products you install. Tailoring involves setting each script's environment variable to point to the directory where you installed the product referenced by the script. If the installation procedure is interrupted before the script tailoring is complete or if the normal installation procedure is not used, you might have to tailor the scripts manually.

Depending on the products you purchase and install, your Progress installation provides the required scripts. These are some of the Progress Version 9.1 startup scripts:

adaptman	mpro	proadsv	prooidrv
aiaman	mssman	proaiw	proserve
asbman	nsman	proapw	proshut
bpro	odbman	probiw	prowdog
dbman	oraman	probrkr	rfutil
mbpro	pro	prooibrk	wtbman

NOTE: The scripts listed above are located in `$DLC/bin`.

If the automatic tailoring did not take place, you receive the following error message when you try to start your Progress product:

```
module-name not found
```

The *module-name* is the Progress module that the script is trying to start. For example, if the script is pro, the module name is _progres.

If it is necessary to tailor your startup scripts manually, follow these steps:

1 ♦ Use any text editor to edit the scripts.

2 ♦ Look for the following syntax:

SYNTAX

```
env_variable=${env_variable-pathname}; export env_variable
```

For example:

```
DLC=${DLC-/usr/grp/dlc};export DLC
```

3 ♦ Change the pathname to the full pathname of the directory where you installed your Progress product.

PART II

Configuration

Setting Up Your Environment

Memory and System Configuration

Configuring Progress

Configuring Progress Unified Broker Products

Maintaining Two Versions

Starting and Running Progress

Terminals

Progress National Language Support

The Progress Directory Structure

Progress Version 9 Typical and Complete Installation Components

5

Setting Up Your Environment

This chapter explains how to run Progress Version 9.1D.

Specifically, this chapter contains the following sections:

- Setting Environment Variables
- UNIX Environment Variables
- Setting Java Environment Variables
- Setting SQL Client Environment Variables
- Using the Open Client Toolkit and the Just-in-Time Compiler
- Running Progress Applications
- Using the Proenv Utility
- Using the AdminServer

NOTE: Before you continue, read the readme file (`README.pro`) located in your installation directory. This text file contains the latest information about Version 9.1D, which might not be included in these installation notes.

5.1 Setting Environment Variables

By default, the Progress installation program tailors all the necessary Progress and Java environment variables to where they are installed. After installing Progress, you can set environment variables to suit your own preferences.

CAUTION: Although editing environment variables is an option, this procedure is not recommended if Progress Version 8.3 and Version 9 (or WebSpeed Version 2.x and Version 3.x) products exist on the same system.

For more information on environment variables for Progress, see the "Maintaining User Environments" chapter in the *Progress Client Deployment Guide* or your specific product documentation.

5.2 UNIX Environment Variables

This section describes the operating system-specific environment variables for the UNIX operating system.

For information about setting environment variables related to Progress AppServer, Progress WebSpeed, a Progress DataServer, or the Progress SonicMQ Adapter, see the AppServer, WebSpeed, DataServer, or *Progress on the Web* documentation.

After installation, Progress requires little additional configuration. There are, however, several environment variables that provide for customization and flexibility. For example, the PATH environment variable allows scripts and images to be found easily, and is the only environment variable needed by most UNIX users.

Typically, the default values of Progress environment variables are based on the `DLC` top level variable. If so, `DLC` must be set to the directory where Progress is installed.

Running the `proenv` script sets `DLC` to this directory automatically. `Proenv` also adds `$DLC/bin` to your path and changes your current directory to the Progress work directory set during installation.

You can edit the `.profile` of a user to set up environment variables automatically each time the user logs onto the system. Also, be sure to export environment variables to make them available to child processes.

This section describes the purpose of each environment variable. Usage with the Bourne shell is given, yet other shells use similar syntax.

Table 5–1 describes the UNIX environment variables. Read these descriptions to determine the variables you want to set.

NOTES

- $DLC is an environment variable for the full pathname of the directory where Progress is installed. You can run Proenv to automatically set DLC to this directory.
- If you want to use a remote DataServer, you must set additional environment variables depending on the type of DataServer you want to use (for example, ORACLE or ODBC). See the DataServer documentation for more information on the other variables set.
- When you first execute a Progress command or utility that requires Java, Progress correctly sets the Java environment variables based on your UNIX platform.

Table 5–1: UNIX Environment Variables *(1 of 5)*

Variable	Description	Code Example
DLC	The pathname of the directory where you installed the Progress software. The default value is /usr/dlc. You must set this variable if you install the Progress software in an alternate directory.	DLC=/usr/dlc
PATH	A list of the directories UNIX searches to find any commands that you provide. Progress also searches these directories for UNIX commands or programs you name when using the INPUT THROUGH and OUTPUT THROUGH statements. Include $DLC/bin in the PATH environment variable. To keep end users out of the /DLC directory, you can provide scripts to perform all Progress-related actions. These scripts can reside somewhere else in the PATH and invoke Progress commands with full pathnames. Place your startup, shutdown, and maintenance scripts somewhere in the path directories. **NOTE:** If during installation you chose **yes** to copy scripts to /usr/bin, just ensure that PATH is set to /usr/bin: PATH=:/usr/bin.	PATH=$PATH:$DLC/bin

Table 5–1: UNIX Environment Variables *(2 of 5)*

Variable	Description	Code Example
PROCFG	The filename (or full pathname) of your product's configuration file. The configuration file is a data file that identifies the Progress product and components that you are licensed to use. The default value is `$DLC/progress.cfg`. Reset PROCFG if you have moved your configuration file from the directory where you installed Progress.	`PROCFG=$DLC/products.cfg`
PROCONV	The filename (or full pathname) of the Progress `convmap.cp` file. The `convmap.cp` file is a binary file that contains all of the conversion tables that are available to Progress. The default value is `$DLC/convmap.cp`. See the *Progress Internationalization Guide* for more information on the `convmap.cp` file.	`PROCONV=$DLC/convmap.cp`
PROEXE	The pathname of your Progress executable file. The default value is `$DLC/bin/_progres`. If you move `_progres` out of `$DLC/bin`, rename `_progres`, or use the PROBUILD utility to generate a customized module, set PROEXE appropriately (or modify your scripts).	`PROEXE=$DLC/bin/_progres`
PROLOAD	The pathname of the directory where you installed the PROBUILD product, if you installed it. The default value is `$DLC/probuild`. For example, if you installed PROBUILD in the directory `/vol1/devdir/load`, use the code example.	`PROLOAD=/vol1/devdir/load`
PROMSGS	The full pathname of your Progress run-time messages file. The default value is `$DLC/promsgs`. For example, if you want to use the German run-time messages file, use the code example in your profile. You only set the PROMSGS environment variable if you want to use a run-time messages file different from the default PROMSGS file in the `$DLC` directory.	`PROMSGS=$DLC/prolang/ger/promsgs.ger`

Table 5–1: UNIX Environment Variables *(3 of 5)*

Variable	Description	Code Example

PROPATH	A list of directories Progress searches to find procedures.
	
	Progress AppServer and Progress WebSpeed use the PROPATH property in `$DLC/properties/ubroker.properties`.
	
	Otherwise, by default, Progress searches:
	1. `$DLC/tty`
	2. `$DLC`
	3. `$DLC/bin`
	
	Use the following syntax to set the `PROPATH` environment variable:
	
	`PROPATH=[:]{`*`dir-name`*`|$`*`env-var`*`}(:...)`
	
	where:
	
	`:`
	Tells Progress to search your working directory before searching any other directories.
	
	`dir-name`
	Specifies the name of a directory that you want Progress to search.
	
	`env-var`
	Specifies the environment variable whose definition names one or more directories that you want to search.
	
	`(:...)`
	Separates multiple *`dir-name`* or *`env-var`*.
	
	`;`
	Ends the definition of the PROPATH environment variable and indicates the start of a new command.
		`PROPATH=:persapp:$DLC`

Table 5–1: UNIX Environment Variables *(4 of 5)*

Variable	Description	Code Example
PROSRV	The pathname of your executable PROSERVE file. The default value is `$DLC/bin/_mprosrv`. The PROSERVE script includes the code example. Therefore, if you move `_mprosrv` out of `$DLC/bin`, rename `_mprosrv`, or use the PROBUILD utility to create a customized module, set PROSRV appropriately (or modify your proserve script).	`PROSRV=$DLC/bin/_mprosrv`
PROSTARTUP	The pathname of the Progress default startup parameter file, `startup.pf`. This file is read by all Progress modules at startup; it must exist for Progress to execute properly.	`PROSTARTUP=$DLC/startup.pf`
PROTERMCAP	The full pathname of the terminal definition file that you want your Progress session to use. The default terminal definition file is called `PROTERMCAP`, and is installed by default in the `/$DLC` directory. You only have to set the PROTERMCAP environment variable if you want to use a different terminal definition file from the default PROTERMCAP file.	`PROTERMCAP=$DLC/SPECIALCAP`
TERM	The type of terminal you are using. For example, to define your terminal type as wy75, use the code example.	`TERM=wy75`
JDKHOME	Establishes the top-level directory for the Java Developer's Kit (JDK). **NOTE:** When you first execute a Progress command or utility that requires Java, Progress correctly sets the Java environment variables based on your version of UNIX.	`JDKHOME=$DLC/jdk`
JREHOME	Establishes the top-level directory for the Java Runtime Environment (JRE).	`JREHOME=$DLC/jre`
JFCHOME	Establishes the top level directory for the Java Foundation Classes (JFC).	`JFCHOME=$DLC/jfc`
JDKCP	Sets the classpath for `class.zip`; Java Developer's Kit (JDK) only.	`JDKCP=$JDKHOME/lib/class.zip`

Table 5–1: UNIX Environment Variables *(5 of 5)*

Variable	Description	Code Example
JRECP	Sets the classpath for Java Runtime Environment (JRE); if no JRE, then it sets classpath for JDK.	`JRECP=$JREHOME/lib/rt.jar`
JFCCP	Sets the classpath for Java Foundation Classes (JFC) only.	`JFCCP=$JFCHOME/swingall.jar`
PROGRESSCP	Contains a list of paths, jar files, and zip files for running Progress-specific products.	`PROGRESSCP=$DLC/java/ progress.zip`
CLASSPATH	Progress correctly sets the appropriate classpath variable based on the platform in use.	`CLASSPATH=$JDKCP:$JFCCP:$PROGRES SCP`
JIT	Sets the just-in-time compiler correctly.	`JIT="-nojit"`
JVMEXE	Sets the Java Virtual Machine to run correctly.	`JVMEXE=jre`

5.3 Setting Java Environment Variables

Progress Version 9 bundles the JRE component with each deployment product you install on the Sun Solaris and HP-UX platforms. By default, the installation program tailors all the necessary Java environment variables to where they are installed.

Additionally, Progress Version 9 bundles the JDK component with each development product that you install on the Sun Solaris platform. For the HP-UX platform, Progress installs the required version of the JRE. If you want access to the JDK, you must install the JDK and set the JDK environment variables. For information about the location of the JRE and JDK for the HP-UX platform, see Chapter 1, “Installation Requirements.”

For the Compaq Tru64, SCO OpenServer 5, and Linux platforms, you must install the required version of either the JRE or the JDK (if not already installed). You must also set JDK environment variables. For information about the location of the JRE and JDK for the Compaq Tru64 platform, see Chapter 1, “Installation Requirements.”

5.3.1 Setting the JDK Environment Variable

To correctly set up your JDK environment for products that rely on the environment variables set by the script file `$DLC/bin/java_env` (for example, JDKHOME and JDKCP), you must edit this file and change the JDKHOME value from:

```
#JDKHOME=
```

to:

```
JDKHOME=/usr1/jdk-directory
```

where `/usr1/`*`jdk-directory`* is the JDK install directory.

NOTE: This modification applies to the following sections of the `$DLC/bin/java_env` file: HP-UX and Compaq Tru64. The modification does not apply to SunOS, because the JDK is bundled on the Sun Solaris platform. Root owns the `java_env` file, and the individual modifying the file must have root access.

For more information on Java environment variables for Progress, see your specific product documentation.

5.4 Setting SQL Client Environment Variables

In order to successfully run a JDBC-based client application, you must set the system shared library search path to include the Progress run-time shared library (`$DLC/lib`) as follows:

On Solaris, Compaq Tru64 UNIX, and Linux:

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$DLC/lib
```

On HP-UX:

```
SHLIB_PATH=$SHLIB_PATH:$DLC/lib
```

On AIX:

```
LIBPATH=$LIBPATH:$DLC/lib
```

The commands for the Progress AdminServer (proadsv), Database Manager (dbman), and SQL Explorer (sqlexp) currently set the necessary directories on the shared library search path for both Progress and Java. You can use these as a reference for your own applications.

5.5 Using the Open Client Toolkit and the Just-in-Time Compiler

If you are using the Java-based Open Client Toolkit and notice frequent core dumps, you might need to disable the Java Just-in-Time (JIT) compiler (by using the `-nojit` option).

5.6 Running Progress Applications

Before running Progress applications, enter this command:

```
cd working-directory
```

CAUTION: Never run Progress products from the directories in which you installed them. Doing so could result in changes to the software that affect its proper operation.

Start a Progress session, and then connect to a database. Choose one of the following:

- To connect to a Progress database from an SQL-92 client application, use SQL Explorer, the JDBC interface, or an application program with embedded SQL-92 statements. SQL Explorer is a Java-based application that allows you to connect to a Progress database through the JDBC interface.

  For more information, see the *Progress JDBC Driver Guide*.

  For complete descriptions of the CONNECT, SET CONNECTION, and DISCONNECT SQL-92 statements, see the "SQL-92 Statements" section in the *Progress SQL-92 Guide and Reference*.

  For information on how to use SQL-92 connection management statements in a C Language application program, see the "Connection Management in ESQL-92" section in the *Progress Embedded SQL-92 Guide and Reference*.

- To connect to a 4GL database, use the pro, bpro, or mbpro command; or use the CONNECT statement. If you are connecting from a Windows client, you can use the Client icon or the PROWIN32 executable. For more details about the commands or the CONNECT statement, see the *Progress Language Reference* or the *Progress Programming Handbook*. For information about the Client icon, see the *Progress Installation and Configuration Guide Version 9 for Windows*.

5.7 Using the Proenv Utility

The $DLC environment variable is not set at the system level. The Proenv utility can automatically set the $DLC environment variable to the directory where Progress is installed. It then adds `$DLC/bin` to your PATH.

Proenv opens a new window, sets the environment variables, and then changes the current directory to the working directory you set when you installed Progress.

5.8 Using the AdminServer

An AdminServer is installed on every system where you install a Progress database, DataServer, AppServer, SonicMQ Adapter, NameServer, or WebSpeed Transaction Server. The AdminServer must be running to use any of the following command-line configuration utilities:

- ADAPTMAN and ADAPTCONFIG SonicMQ Adapter utilities
- AIACONFIG AppServer Internet Adapter utility

  NOTE: There is no AIAMAN command, because the Web server's Java servlet engine starts it. You can control the AppServer Internet Adapter through the Progress Explorer. See the chapter on the AppServer Internet Adapter in *Progress on the Web* for more information.

- ASBMAN and ASCONFIG AppServer utilities
- DBMAN database utility
- MSSCONFIG and MSSMAN Microsoft® SQL Server™ DataServer utilities
- NSCONFIG and NSMAN NameServer utilities
- ODBCONFIG and ODBMAN ODBC DataServer utilities
- ORACONFIG and ORAMAN DataServer utilities
- WSCONFIG and WTBMAN WebSpeed utilities

Supporting Progress Version 9.1 administrative capabilities on UNIX is a command-line utility, `proadsv`, which allows you to start up and shut down the AdminServer.

NOTE: Before you start a WebSpeed or AppServer application, you must start the AdminServer. In Version 9.1D, the AdminServer User-Group Authorization feature has been introduced. It requires that you have privileges set to allow you access and operational privileges for the AdminServer. See the "How To Implement the User-Group Authorization Feature," section later in this chapter.

The AdminServer must be running before you can use the Progress Explorer, from a remote Windows machine, to configure and manage your applications.

5.8.1 proadsv

The proadsv utility is used to start up or shut down the current installation of an AdminServer. This is the syntax for proadsv:

SYNTAX

```
proadsv { { { -start { [ -adminport port-number ] }
                | -stop | -query } [ -port port-number ] } | -help }
```

PARAMETERS

`-start`

Starts the AdminServer.

`-adminport` *`port-number`*

Specifies the port number used by the AdminServer for database broker communication. If a port number is not specified, the adminport defaults to port 7832.

`-stop`

Stops the AdminServer.

`-query`

Displays the AdminServer status.

`-port` *`port-number`*

Specifies the listening port number. If a port number is not specified, the port defaults to 20931.

`-help`

Displays the command-line help.

Table 5–2 shows several options that you can use with `proadsv` to accomplish the corresponding tasks. Note that the examples use the port number `9999`.

Table 5–2: proadsv Command-line Options

AdminServer Task	Commands	Examples
Start	`-start`	`proadsv -start`
Specify the listening port	`-port` *`port-number`*	`proadsv -port 9999 -start`
Specify the database broker port	`-adminport` *`port-number`*	`proadsv -adminport 9998`
Stop	`-stop`	`proadsv -stop`
Query	`-query`	`proadsv -query`
Help	`-help`	`proadsv -help`

NOTES

- The port numbers specified with the `-port` and `-adminport` options must be different.
- If you are running multiple AdminServers you need to override both the default port and the default adminport settings.

5.8.2 How To Implement the User-Group Authorization Feature

To implement the User-Group Authorization feature on a UNIX platform, you must first successfully complete the Version 9.1D installation program.

Table 5–3 identifies and briefly describes the purpose of each new command-line option:

Table 5–3: User-Group Parameter Options

Parameter name	Syntax	Purpose
Individual user name and password required	`-requireusername`	To require a minimum of one user ID to be resolved for each AdminServer operation before it can be executed.
Group authorization required	`-admingroup group` [{,|:}group...]	To require a minimum of one group to be resolved for each AdminServer operation before it can be executed. On a UNIX platform, a comma-separated list differentiates groups when you are specifying multiple groups on the command line.

On UNIX platforms, a group name can be any user-defined or NIS group name. UNIX can also support subgroups.

6

Memory and System Configuration

This chapter describes how to manage your system's memory and configuration. The chapter also lists the Progress installation subdirectories and their contents.

Specifically, this chapter includes the following sections:

- Calculating Memory Needs
- Managing Shared Memory and Process Resources
- Reducing Memory Usage
- Swap Space
- Shared Memory and Kernel Configuration

6.1 Calculating Memory Needs

The tables in this section are provided to help you calculate the memory requirements for your system. Table 6–1 lists the components you use to calculate system memory requirements. Table 6–2 lists the startup options that affect memory requirements.

Table 6–3 and Table 6–4 list approximate values for each calculation component for single and multiple users running Progress. Use the formulas provided in Table 6–5 to calculate the memory requirements for your system without disk swapping.

NOTE: All memory usage figures are approximate and vary depending on the version of the operating system, UNIX parameters, the Progress startup parameters, and the Progress application you are using. For more information, see the *Progress Startup Command and Parameter Reference*.

The background processes APW, BIW, AIW, and PROWDOG also take up memory. Remember to calculate these in your memory requirements.

Table 6–1: Components Used To Calculate Memory Needs *(1 of 2)*

Component	Symbol	Comment
Operating System	os*	Represents the memory requirements for one copy of your operating system shared in memory by all users, plus a certain percentage of physical memory to allow for operating system buffers; typically, the percentage is 10%-15%.
Progress	_progres*	Represents the size of one copy of Progress shared in memory by all users running single-user or multi-user Progress. Allow for 15%-20% deviation in the _progres value to accommodate new releases.
Database server or broker	_mprosrv*	Represents the size of one copy of the Progress database broker/server shared in memory by all users running multi-user Progress. Use this component only when calculating memory requirements for a system running a multi-user version of a Progress product.

Table 6–1: Components Used To Calculate Memory Needs *(2 of 2)*

Component	Symbol	Comment
Progress user data	proud	Represents the data area required for each user running Progress. This value varies greatly, depending on the application you run and whether you use the Progress Compiler. It is also affected by many of the startup parameters. For single-user clients, the parameters are: Blocks in Database Buffers (-B), Directory Size (-D), and Stack Size (-s). For multi-user clients, the parameters are: -D, Maximum Memory (-mmax), and -s.[1,2]
Progress server data	psd	Represents the data area required for each database server serving remote clients. (Not used for single-user or multi-user clients if the users are self-service). This space is used for communication buffers and other server memory requirements.
Progress broker data	pbd	Represents the data area required by each database broker. (One database broker is required for each different database simultaneously in use in multi-user mode whether you are using remote client/servers, self-service, or both.) This value is determined by the values of startup parameters that consume memory, including: Database Buffers (-B), Lock-table Entries (-L), and Number of Users (-n).[2] **NOTE**: Each increment of -n increases pbd by 2K.

1 Use the UNIX `size` command to determine the exact size. See Table 6–2 to determine the approximate value.

2 See the *Progress Startup Command and Parameter Reference* for information about Progress startup parameters.

Table 6–2: Size Increments For Increasing Startup Parameters By 1 *(1 of 2)*

Startup	Size Increment	Affects
Blocks in Database Buffers (-B)	db block size (.5K, 1K, 2K, 4K, 8K)	multi-user: pbd; single-user: proud
Directory Size (-D)	100 bytes	proud
Lock-table Entries (-L)	16 bytes	pbd

Table 6–2: Size Increments For Increasing Startup Parameters By 1 *(2 of 2)*

Startup	Size Increment	Affects
Shared-memory Size (-Mxs)	1K	pbd
Number of Users (-n)	2K	pbd
Stack Size (-s)	1K	proud

Table 6–3: Single-user Memory Requirements

Component Symbol	Memory
`_progres`	3.5MB-4MB[1]
`proud`	1MB-2MB[2]
`Kernel`	3MB-5MB[3]

1 This is an approximate value. Use the `size` command to determine the exact size. If you are using a non-Progress database, your value will be larger.

2 The actual value depends on your application and how you set the startup parameters listed in Table 6–2.

3 Use the UNIX `size` command to determine the exact size of your kernel. Add to that 10% of physical memory on the machine, as an allowance for operating system buffers, to get the total operating system requirements.

Table 6–4: Multi-user Memory Requirements *(1 of 2)*

Component Symbol	Memory
`_progres`	3.5MB-4MB[1]
`_mprosrv`	.5MB-.55MB[1]
`proud`	1MB-3MB[2]
`Kernel`	.3MB-.5MB[3]
`psd`	.5MB-.55MB[1]

Table 6–4: Multi-user Memory Requirements *(2 of 2)*

Component Symbol	Memory
pbd	psd + -B, etc.[2]

1 This is an approximate value. Use the `size` command to determine the exact size. If you are using a non-Progress database, your value will be larger.

2 The actual value depends on your application and how you set the startup parameters listed in Table 6–2.

3 Use the UNIX `size` command to determine the exact size of your kernel. Add to that 10% of physical memory on the machine, as an allowance for operating system buffers, to get the total operating system requirements.

Table 6–5: Formulas For Calculating Memory Requirements

Single-user Systems	Multi-user Systems
os + _progres + (number of users x proud)	os + _progres + _mprosrv + (number of databases x pbd) + (number of remote client servers x psd) + (number of users x proud)

NOTE: Remote client/server processes share the same code as the broker and, therefore, require no additional _mprosrv (database server or broker) memory. Each remote client/server process does require a Progress server data (psd) area.

6.2 Managing Shared Memory and Process Resources

Progress uses several operating system resources, such as shared memory and memory locks, processes, and client memory. You can plan Progress operations more effectively if you understand these resources.

6.2.1 Shared Memory

Shared memory is an area in system memory that multiple users can access concurrently. Progress stores shared resources in the shared-memory area; more than one user and server can access each database. Progress uses semaphores and spin locks to synchronize the activities of server and self-service client processes that are connected to a database. Each process uses its semaphore or relies upon the spin lock when it must wait for a shared resource.

You can tune Progress performance by reconfiguring the size of the following shared-memory buffers:

- **Database buffers** — Progress reads database blocks into the database buffer pool. Larger buffers usually result in less disk I/O.
- **Before-image (BI) buffers** — Progress stores BI notes in memory before writing them to disk.
- **After-image (AI) buffers** — Progress stores AI notes in memory before writing them to disk.

Progress also creates shared-memory tables to provide essential information on the status of each process, server, transaction, and lock. These tables enable you to control all of the database activities from one shared area.

See the *Progress Database Administration Guide and Reference* for more information about improving performance.

6.2.2 Processes

Progress provides the following optional processes to improve performance:

- **Asynchronous Page Writer (APW)** — The APW improves database performance by performing overhead operations in the background. These operations provide available buffers, reduce the number of buffers that Progress reads before writing to disk, and reduce the overhead associated with before-image checkpointing (the process of synchronizing the buffer pool of modified blocks to the database).
- **Before-image Writer (BIW)** — The BIW improves performance by continually writing before-image buffers to disk. These writes occur in the background.
- **After-image Writer (AIW)** — The AIW improves performance by continually writing AI buffers to disk soon after Progress fills the buffers.
- **Progress Watchdog (PROWDOG)** — The watchdog process cleans up after improperly terminated processes by releasing locks, backing out any live transactions and releasing shared-memory locks, and disconnecting and cleaning up the server's remote clients.

See the *Progress Database Administration Guide and Reference* for details on using processes to improve Progress performance.

6.3 Reducing Memory Usage

If you run Progress and find there is not enough main memory, try the following to reduce main memory use:

- Reduce the amount of memory allocated to Progress database buffers, as controlled by the -B startup parameter.
- Change other startup parameters, such as -n and -L.

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

6.4 Swap Space

When the amount of memory used by all processes running on a UNIX system exceeds the amount of physical memory, portions of memory are swapped to disk. A special area of the disk is reserved for this swapping. The system administrator can set the size of this area when configuring the system.

NOTE: Progress Software Corporation recommends that you set your swap space size to at least twice the size of your system memory.

A UNIX system can deadlock while accessing the disk when the swap space is used up. This can happen when too many large processes are running simultaneously. If you expect to have a larger than normal number of users, or if Progress memory requirements are larger than your typical process, consider increasing the amount of swap space available on your system. Before you change the size of the swap area, back up and reformat the disk.

The UNIX user set-ID bit is turned on for the Progress program module. Consequently, even though there might be no active Progress users, this module remains in the UNIX swap area on disk until you shut down the system.

6.5 Shared Memory and Kernel Configuration

This section applies to shared-memory UNIX systems. Most UNIX systems support shared memory and semaphores; however, if your system does not support them, disregard this section.

In Progress, the multi-threaded architecture makes heavy use of file descriptors, shared memory, and semaphores. Allocation of these resources is controlled by system configuration parameters. On most systems, these parameters are set to values appropriate for Progress applications. However, in some cases, one or more parameters might not be set optimally, thereby limiting the number of Progress users. If you have to reset the parameters, you must edit the parameter file and reconfigure your kernel. The kernel reconfiguration procedure is commonly documented in the system administration manual provided by your hardware vendor.

The optimal parameter settings depend on the system, the application, the number of users, and some minor factors. Table 6–6 lists the crucial parameters and provides guidelines for choosing adequate values for each one.

Table 6–6: Shared Memory and Semaphore Parameter Settings

Parameter	Meaning	Optimal Setting
SHMMNI	Maximum number of shared memory (SHM) identifiers.	2 per active multi-user database; if (#users > 50), add 1 per 25 users; if (-B > 400), add 1 per 200 buffers.
SHMSEG	Maximum number of SHM segments a single process can attach.	4-8
SHMALL	Maximum number of in-use SHM segments.	System default; increase if many databases are active simultaneously; decreasing -B, -n, and -L startup parameters decreases SHM requirements.
SHMMAX	Maximum SHM segment size.	System default; increase if you get Progress error 1135.
SEMMNI	Number of semaphore (SEM) IDs; each represents an array of SEMs.	1 per active multi-user database.
SEMMSL	Maximum number of semaphores per SEM ID.	(Max-local-users-on-any-database + Max-#servers-on-any-database + 4).
SEMMNS	Total semaphores in the system.	(SEMMSL x #active-databases).
SEMMNU	Number of semaphore undo structures.	Same value as SEMMNS.
MAXUMEM	Maximum address space for a single user.	> = server size process + SHMSEG * SHMMAX.

The parameter settings in Table 6–6 are guidelines. Parameter values near these are acceptable in most cases, but a particular system or application might require increasing the limits.

If shared memory or semaphores are allocated incorrectly, Progress displays an error message when it attempts to start an additional user or server. For example, if SEMMNS is set too low, PROSERVE fails and displays the following message:

```
Server: Semaphore limit exceeded
Server: **The server terminated with exit code (X) (800)
```

Change the relevant parameter values and reconfigure the kernel in response to semaphore or shared-memory errors at startup. Table 6–7 lists the parameters that you might have to raise in response to various Progress error codes.

Table 6–7: Error Codes and Kernel Reconfiguration Parameters

Error Code	Parameter to Increase
1081	SEMMNU
1093	SEMMSL or SEMMNS
1130	SEMMSL
1131	SEMMNI and SEMMNS
1135	SHMMAX, MAXUMEM, and MAXUP
1137	SHMMNI
1175	SHMSEG, MAXUMEM, and MAXUP
1195	SEMMNS

NOTE: The Blocks in Database Buffers (-B), Lock-table Entries (-L), and Number of Users (-n) startup parameters all affect shared-memory usage. The Number of Users (-n) and Maximum Servers (-Mn) parameters affect semaphore usage (each user or server process uses one semaphore). Before reconfiguring your kernel to increase shared memory or semaphore allocation, see whether you can lower these startup values.

7

Configuring Progress

This chapter provides information about shared-memory configurations, multi-tier configurations, client/server configurations, and preparing to run Progress on a TCP/IP network.

Specifically, this chapter contains the following sections:

- The Progress Explorer Framework
- Shared-memory Configurations
- Multi-tier Configurations
- Client/Server Configurations
- Client/Server and Progress AppServer In the Network Environment
- Preparing To Run Progress In a TCP/IP Network

7.1 The Progress Explorer Framework

The Progress Explorer Framework is a system administration framework that provides a consistent interface for managing all Progress products installed on your network. The framework consists of the following elements:

- **AdminServer** — Provides secure administrative access to Progress products
- **Progress Explorer Configuration Tool** — A Windows-based configuration tool with graphical user interfaces that you can use for remote system administration

  NOTE: For more information about the Progress Explorer, see the chapter on configuration in the *Progress Installation and Configuration Guide Version 9 for Windows* or the Progress Explorer online help.

- **Command line configuration utilities** — Character versions of the Progress Explorer configuration tools

7.1.1 AdminServer

An AdminServer is installed on every system where you install a Progress database, DataServer, AppServer, SonicMQ Adapter, NameServer, or WebSpeed Transaction Server. The AdminServer must be running in order to use the command-line configuration utilities or Progress Explorer configuration tools, from a remote Windows machine, to manage your Progress installation. The AdminServer grants access to each instance of an installed Progress product.

For more information about running the AdminServer and for the complete syntax of the PROADSV, utility see the "Using the AdminServer" section in Chapter 5, "Setting Up Your Environment."

7.1.2 Command-line Configuration Utilities

The command-line configuration utilities allow you to start, stop, and configure installed Progress products. The command-line configuration utilities that are part of the Progress Explorer Framework include:

- **ADAPTMAN** — Helps manage the operation of a configured SonicMQ Adapter
- **ADAPTCONFIG** — Helps configure an existing SonicMQ Adapter
- **AIACONFIG** — Helps configure an existing AppServer Internet Adapter

NOTE: There is no AIAMAN command because the Web server's Java servlet engine starts it. You can control the AppServer Internet Adapter through the Progress Explorer. See the chapter on the AppServer Internet Adapter in *Progress on the Web* for more information.

- **ASBMAN** — Helps manage the operation of a configured AppServer
- **ASCONFIG** — Helps configure an existing AppServer
- **DBMAN** — Starts, stops, and queries the current configuration of a Progress database
- **MSSMAN** — Helps manage the operation of a configured DataServer for Microsoft SQL Server
- **MSSCONFIG** — Helps configure an existing DataServer for Microsoft SQL Server
- **NSMAN** — Helps manage the operation of a configured NameServer
- **NSCONFIG** — Helps configure an existing NameServer
- **ODBMAN** — Helps manage the operation of a configured DataServer for ODBC
- **ODBCONFIG** — Helps configure an existing DataServer for ODBC
- **ORAMAN** — Helps manage the operation of a configured DataServer for ORACLE
- **ORACONFIG** — Helps configure an existing DataServer for ORACLE
- **PROADSV** — Starts, stops, or queries the AdminServer
- **WTBMAN** — Helps manage the operation of a configured WebSpeed Transaction Server
- **WSCONFIG** — Helps configure an existing WebSpeed Transaction Server or Messenger

For the complete syntax of each utility, see *Building Distributed Applications Using the Progress AppServer*, the Progress DataServer Guides, *Progress on the Web*, and the *Progress Database Administration Guide and Reference*.

For more information about NSMAN and NSCONFIG, see the "The Progress Unified Broker Administration Framework" section in Chapter 8, "Configuring Progress Unified Broker Products."

7.1.3 Managing Progress Product Configurations

The `ubroker.properties` file stores all the configuration definitions for each instance of the following Progress products:

- NameServer
- AppServer
- AppServer Internet Adapter
- DataServer
- SonicMQ Adapter
- WebSpeed Transaction Server

Each configuration definition contains environment variables and property settings for each product instance. The command-line configuration utilities use the `ubroker.properties` file to store, validate, and manage the configurations for the products.

The file consists of a hierarchical structure of configuration entities, where parent entities provide configuration information that you can override or extend in each child entity. Each configuration entity has a name that begins the entity definition, and the definition contains configuration settings for one or more product instances.

For example, the AppServer configurations in `ubroker.properties` can include:

Configuration Entity Name	Configuration Entity Function
`[UBroker]`	Defines default property settings for all NameServer, AppServer, DataServer, and WebSpeed Transaction Server brokers.
`[UBroker.AS]`	Defines default property settings for all instances of an AppServer.
`[UBroker.AS.`*`product-instance-name`*`]`	Defines property settings for this instance of an AppServer. The `ubroker.properties` file can contain several of these entities each with a unique *`product-instance-name`*.

Parent entities provide default values for all of their child entities. However, at any child level, a redefinition of any value supercedes the default value of its parent. All children from the redefinition level down inherit this new value.

The ubroker.properties file is installed in the properties subdirectory of the Progress installation directory.

The conmgr.properties file stores all the properties for each instance of a Progress database.The conmgr.properties file is installed in the properties subdirectory of the Progress installation directory.

Working With Supported Products

For the definitions and usage of all properties supported in the properties file, see the comments in the installed ubroker.properties file. For more information on:

- Each configuration management and validation utility, see the *Progress Client Deployment Guide* and the *Progress Database Administration Guide and Reference*.
- Maintaining ubroker.properties for AppServer installations, see *Building Distributed Applications Using the Progress AppServer*.
- Maintaining ubroker.properties for DataServer installations, see the Progress DataServer Guide for your DataServer product.
- Maintaining ubroker.properties for WebSpeed installations, see the *WebSpeed Installation and Configuration Guide* for your operating system.
- Maintaining ubroker.properties for NameServer installations, see the configuration manual for each product that includes the NameServer in its configuration. Also see the "NameServers" section in Chapter 8, "Configuring Progress Unified Broker Products," for more information on how to configure NameServers and begin configuration and operation of your Unified Broker product.
- Maintaining ubroker.properties for AppServer Internet Adapter installations, see *Progress on the Web*.
- Maintaining ubroker.properties for SonicMQ Adapter installations, see *Progress on the Web*.

Editing and Validating the Content Of the ubroker.properties File

To configure products you must edit the `ubroker.properties` file using a text editor.

To ensure proper run-time processing, the file must be named `ubroker.properties` and must be located in the `properties` subdirectory of the Progress installation directory. Follow these steps to edit the file and create or modify a product configuration:

1 ♦ Make a backup copy from the installed `ubroker.properties` file (named, for example, `test.properties`).

2 ♦ Make changes to `test.properties` with a text editor.

3 ♦ Run the appropriate validation utility on `test.properties` using the -propfile option to validate your changes. Table 7–1 lists the commands to run the validation utility for each supported product. The command you run must have local access to the `ubroker.properties` file whose configurations you want to validate.

Table 7–1: ubroker.properties File Validation Utilities

Product	Utility
AppServer	`asconfig`
AppServer Internet Adapter	`aiaconfig`
DataServer for Microsoft SQL Server	`mssconfig`
DataServer for ODBC	`odbconfig`
DataServer for Oracle	`oraconfig`
NameServer	`nsconfig`
SonicMQ Adapter	`adaptconfig`
WebSpeed	`wsconfig`

4 ♦ Shut down any running brokers or NameServers using the -stop option of the appropriate configuration management utilities (for example, nsman and asbman).

5 ♦ Copy `test.properties` to `ubroker.properties` in the `properties` subdirectory of the Progress installation directory. If you might return to the previous configuration, store a backup copy of the `ubroker.properties` file before replacing it with your modified version.

6 ♦ Restart your brokers and NameServers using the -start option of the appropriate configuration management utilities. Table 7–2 lists the commands to run the validation utility for each supported product.

Table 7–2: ubroker.properties File Management Utilities

Product	Utility
AppServer	`asbman`
DataServer for Microsoft SQL Server	`mssman`
DataServer for ODBC	`odbman`
DataServer for Oracle	`oraman`
NameServer	`nsman`
SonicMQ Adapter	`adaptman`
WebSpeed	`wtbman`

For more information on the ubroker.properties file see the "Unified Broker Properties File (ubroker.properties)" section in Chapter 8, "Configuring Progress Unified Broker Products."

7.2 Shared-memory Configurations

Shared memory is an area in system memory that multiple users can access concurrently. Most UNIX systems use shared memory.

Shared memory systems run a version of Progress that uses the shared-memory architecture to provide superior performance and additional features.

On shared memory systems, more than one server can access a single database, service local or remote clients, and handle a variety of network protocols. Progress keeps resources shared by all database users in shared memory and lets multiple servers access those resources efficiently. Optionally, additional asynchronous I/O processes can offload I/O operations from each server, further improving resource utilization.

Progress uses semaphores and spin locks to synchronize the activities of server and self-service client processes that are connected to a database. Each process uses its semaphore or relies upon the spin lock when it must wait for a shared resource.

You can tune Progress performance by reconfiguring the size of the Database, Before-image (BI), and After-image (AI) buffers. See the "Shared Memory" section in Chapter 6, "Memory and System Configuration," for more information about reconfiguring these buffers.

Local clients running multi-user Progress can access database resources directly, rather than through a database server. This eliminates client/server message exchange and task-switching overhead. Database requests do not have to be queued until a server can process them. Local direct-access clients are known as *self-service clients*.

To run shared-memory versions of Progress over a network, you need information regarding network-related system files, network configuration, and the startup parameters required to start remote clients. For more information about the network files and configuration, see the "Client/Server and Progress AppServer In the Network Environment" section and the "Preparing To Run Progress In a TCP/IP Network" section in this chapter. For information about starting remote clients, see Chapter 10, "Starting and Running Progress."

Figure 7–1 shows a sample shared-memory configuration.

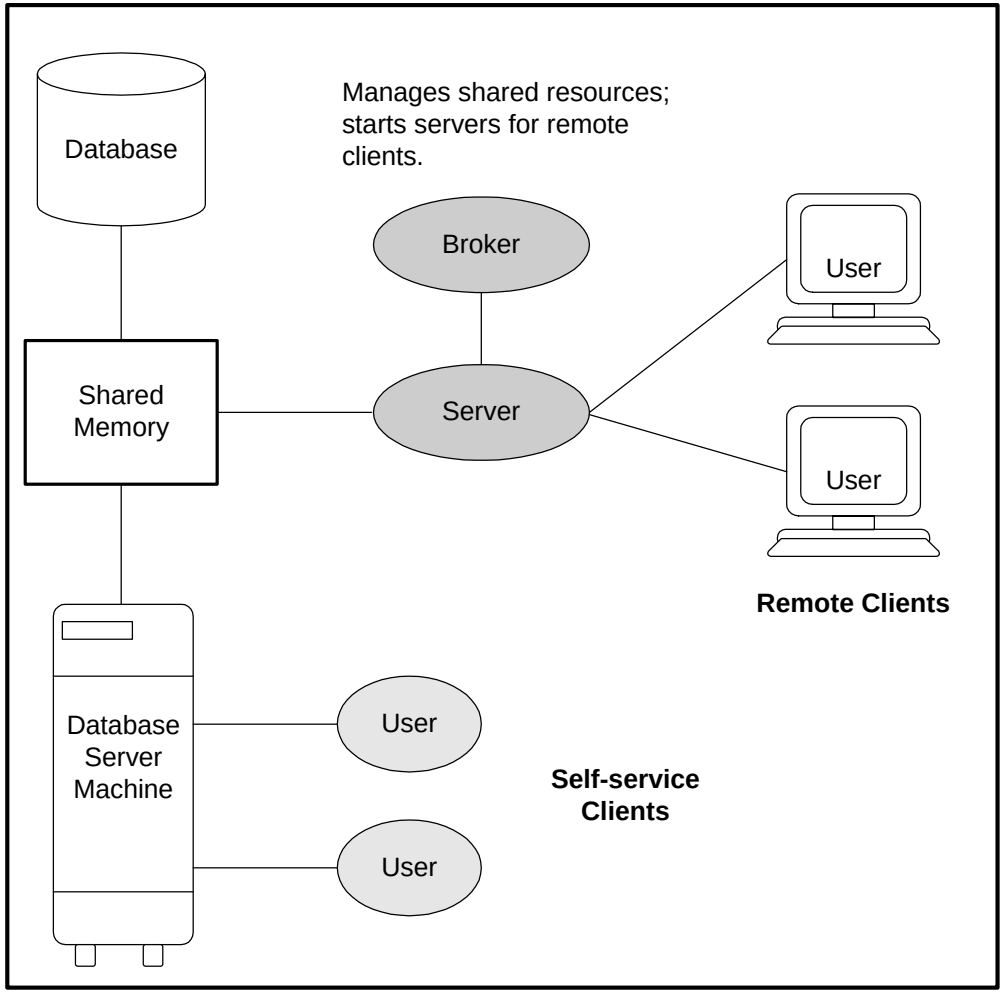


Figure 7–1: Database Access On a Shared-memory System

7.3 Multi-tier Configurations

For large installations, Progress Software Corporation recommends a multi-tier configuration to improve system performance. Figure 7–2 illustrates a three-tier configuration that consists of a database tier that supports self-service clients, an application tier that supports remote clients, and a thin-client tier.

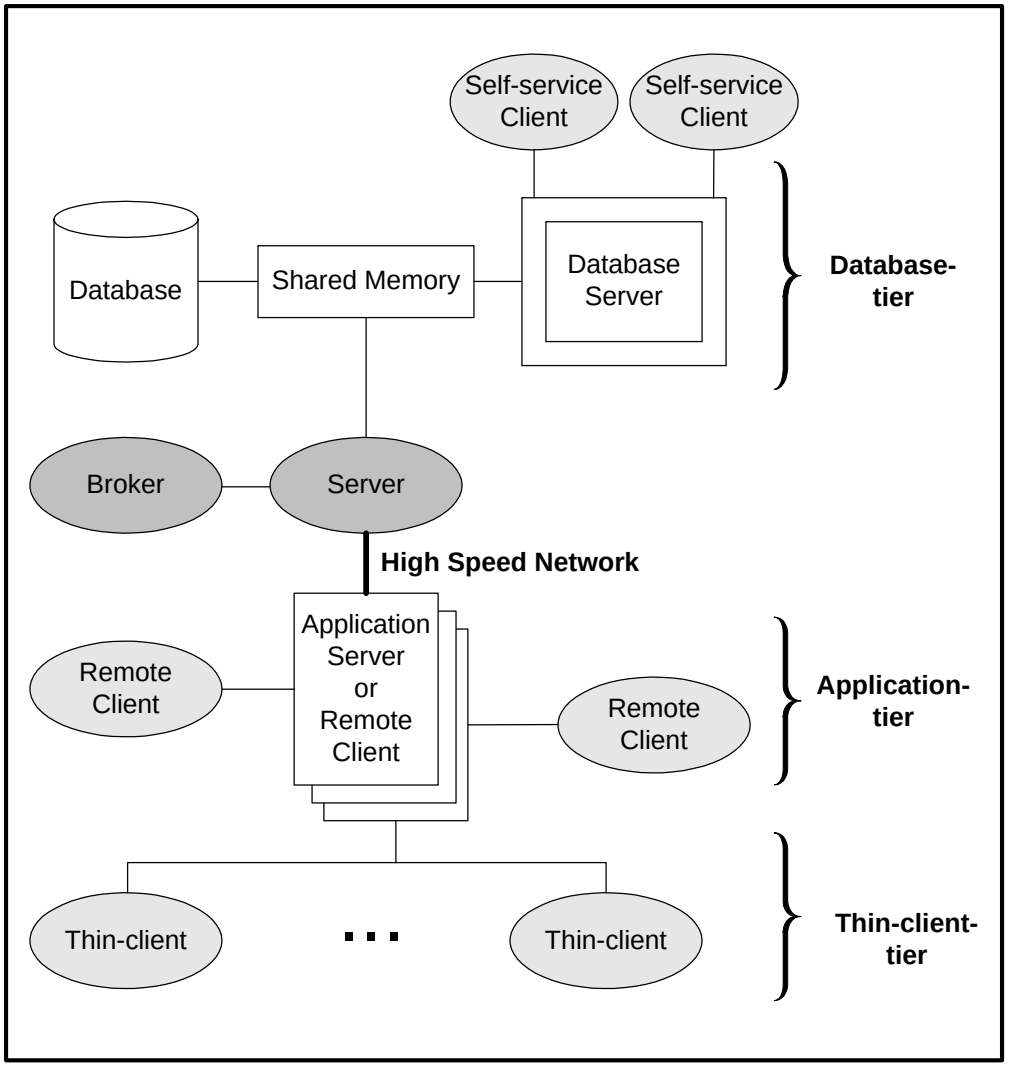


Figure 7–2: Sample Multi-tier Configuration

7.3.1 Shared-memory Architecture

Figure 7–3 shows the shared-memory Progress architecture.

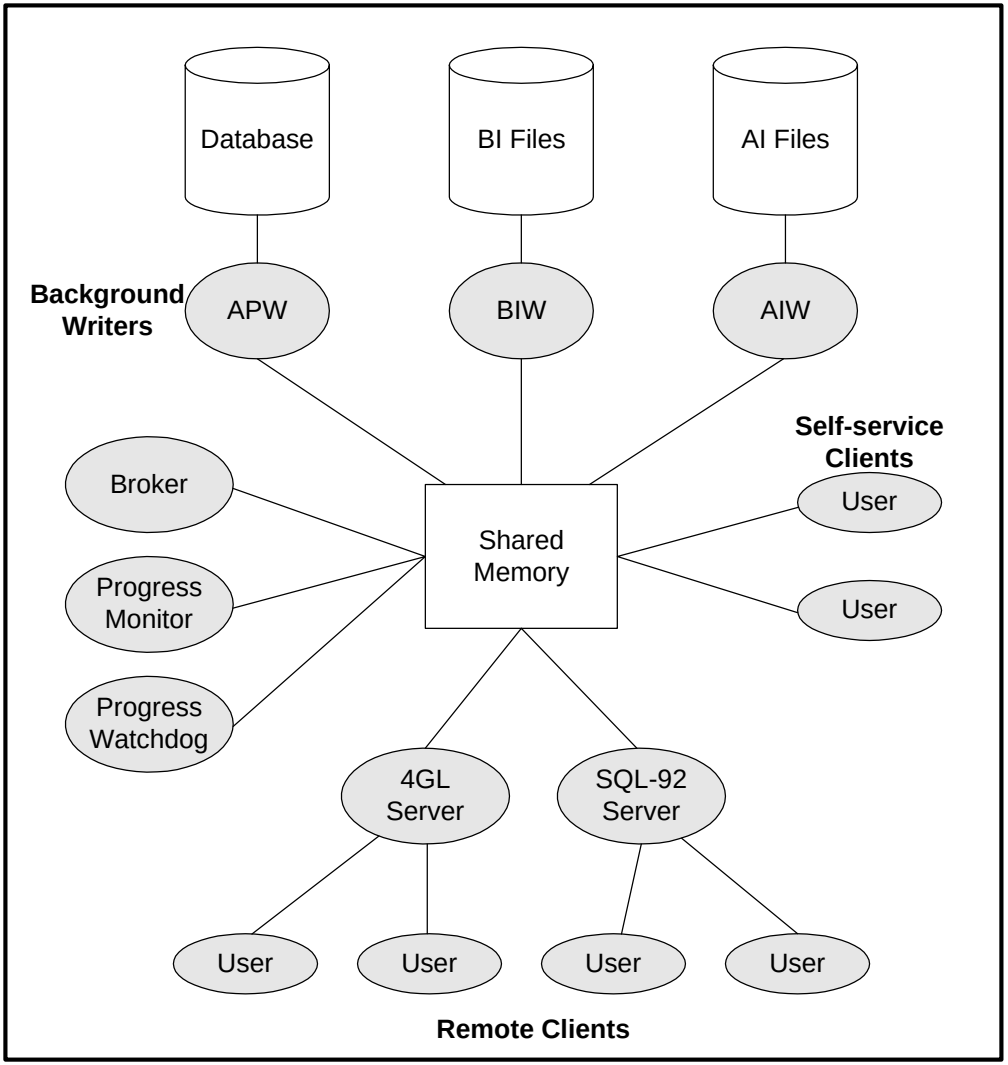


Figure 7–3: Shared-memory Progress Architecture

The following sections explain the components of the architecture that are specific to shared-memory systems.

Progress Database Monitor Utility

The Progress Database Monitor utility displays performance and usage information about database status and activity.

For more information about the Database Monitor utility, see the description of the PROMON utility in the *Progress Database Administration Guide and Reference*.

Progress Watchdog Utility

If a process terminates improperly, it can maintain a lock on a record or shared-memory structure. This can impact database concurrency. The Progress Watchdog utility detects processes that have terminated improperly and cleans up after them.

At regular intervals, the Watchdog utility checks for processes that have terminated unexpectedly. If it finds one, it releases any locks or shared-memory structures that the process might hold.

The Watchdog utility checks for inactive processes approximately once every 10 seconds. It also checks for self-service clients that are no longer active, releases all the appropriate record locks, backs out of any live transactions, and releases any shared-memory locks. If a server process terminates unexpectedly, the Watchdog utility disconnects and cleans up the server's remote clients.

For more information about the Watchdog utility, see the description of the PROWDOG utility in the *Progress Startup Command and Parameter Reference*.

Background Writers

Shared-memory Progress offers three background writer processes that improve performance. These processes continually perform certain housekeeping functions in the background. Because these functions are performed regularly by the dedicated background writer processes, client and server processes rarely have to wait for these functions to be performed.

The three types of background writers (asynchronous page writers, before-image writers, and after-image writers) are described in the "Processes" section in Chapter 6, "Memory and System Configuration."

For more information about background writers, see the *Progress Database Administration Guide and Reference*. You must manually start the background writers. For instructions, see Chapter 10, "Starting and Running Progress."

7.4 Client/Server Configurations

Wherever it runs, multi-user Progress functions in a client/server architecture. On a single machine, Progress provides multi-user access to a database by using a separate client process for each user. Depending on the operating system, each client then accesses the database either directly or indirectly with the help of a server process. While separate and distinct, the Progress client and server processes compete for the same machine resources.

In client/server configurations, the client application and the database server are separate processes. Client processes can be local or remote.

The Progress user interface and Progress applications execute in the client session, sending requests to the Progress server. The Progress server accesses the database on behalf of each client session.

7.4.1 Terminology

This section introduces the terminology used to describe client/server configurations.

Application Workstation

An *application workstation* is any node that runs one or more Progress clients. A workstation running UNIX can run several clients. Depending on its operating system and configuration, an application workstation might run local clients and servers as well.

Database Server Machine

A *database server machine* is any node that runs one or more Progress servers for local or remote Progress clients. Progress Software recommends that you store the database on a disk locally attached to the database server machine. This guarantees the highest level of both performance and database integrity.

Network File Server

A *network file server* is any node that provides shared services such as file, printing, and security services to other nodes, including application workstations and database server machines. It usually provides these services by allowing other nodes to access its local files and printers as if they were local to the other nodes. For example, Progress clients can run application procedures and Progress servers can access database files stored on a remote network file server.

A network file server usually runs the *kernel* of a Network Operating System (NOS), the program that controls access to shared network resources. Depending on its operating system, a network file server might also run one or more Progress database clients and servers.

Although Progress Software recommends that you store the database on a disk locally attached to the database server machine, you can store the database on a network file server. Clients can access shared application code and communicate with the database server. Depending on your application and network environment, however, you might lose database integrity.

Note that Progress often runs in local area networks (LANs) that have no network file servers. On these LANs, application workstations can access only locally stored procedures, and database server machines can access only locally stored databases. However, the application workstations and database server machines can communicate with each other as remote processes.

Single-process Database Server Machine

A *single-process* database server machine is a node that runs only one server process for each database, providing access to that database for self-service clients only.

Multi-process Database Server Machine

A *multi-process* database server machine is a node that runs multiple server processes for each database, providing multiple data paths to the database. Each server queues and runs requests for one or more clients. A separate broker process starts a new server for each additional client (or set of clients, in specified increments) that access the database. All nodes that run operating systems with shared memory run as multi-process database server machines. Progress supports multi-process database server machines on UNIX nodes. For more information, see the "Shared-memory Configurations" section.

You can dedicate all the resources of a database server machine to run database servers. However, depending on your application and operating system, you can also run local clients, as well as remote clients, for other database server machines.

7.4.2 Simple Client/Server Configurations

Figure 7–4 shows a simple client/server configuration, where the client and server components both run on a single system.

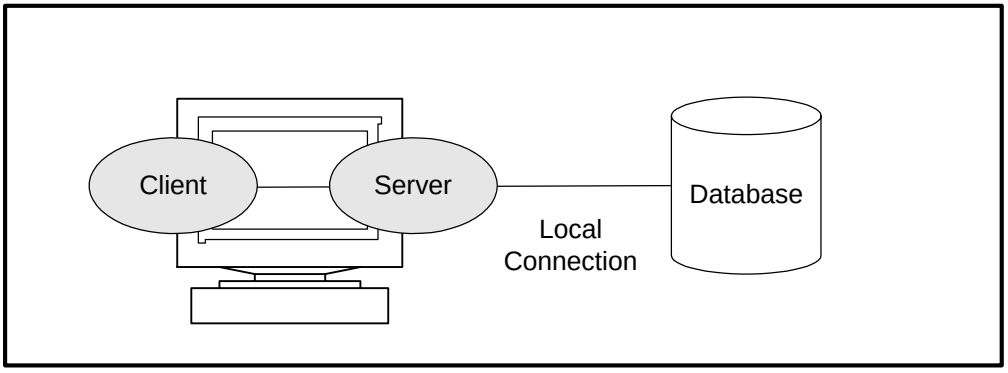


Figure 7–4: Simple Client/Server Configuration

Figure 7–5 shows a multiple system client/server configuration. In this configuration, the server runs on the system where the database resides. The clients run on remote systems, accessing the database through the server system.

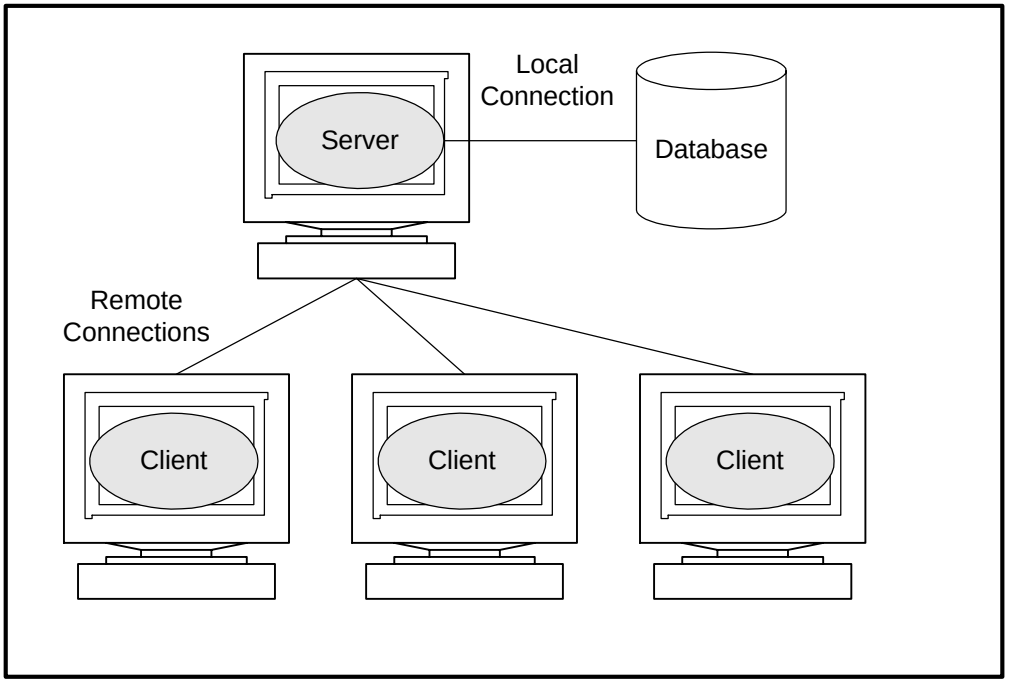


Figure 7–5: Multiple System Client/Server Configuration

7.5 Client/Server and Progress AppServer In the Network Environment

The Progress client/server architecture fits naturally into a network environment, allowing clients and servers to run together in many different (heterogeneous) hardware and operating system environments. On a network, the Progress client and server processes are distributed to separate nodes where they communicate through a common network protocol. Some nodes run client processes, while others run server processes. One advantage of this is that adding users or databases has minimal impact on the machine resources used by others. Each has its own resources devoted only to its client or server tasks. Another advantage is that a single Progress application can take advantage of the strengths of a multi-machine, multi-operating system environment, without regard to differences in file resources on the separate machines. Remote Progress clients and servers interact transparently, regardless of the type of machine environment in which they run. The result is a cooperative application environment with many more possibilities for expansion.

Progress Network Support

Table 7–3 describes the network types that Progress supports.

Table 7–3: Progress Network Type Options For Progress

Network Type (-N)	Description
SNA	Allows some UNIX clients to access a Progress AS/400 server using the APPC protocol suite (Advanced Program to Program Communication using SNA Logical Unit (LU) type 6.2). For more information, see the *Progress/400 Product Guide*.
TCP	Allows client and server operations among UNIX and Windows systems that communicate using TCP/IP. In a Progress AppServer configuration, the client connection to the application server is always TCP. The Progress AppServer supports all of the client/server network types for the connection of the application server to the database server.

Sample Network Configurations

The following figures show sample configurations for Progress on a LAN. Your configuration might differ depending on your requirements.

Figure 7–6 shows the simplest Progress network configuration: a database server machine and application workstation. Although the figure shows only one database server machine and workstation, there can be more than one of each.

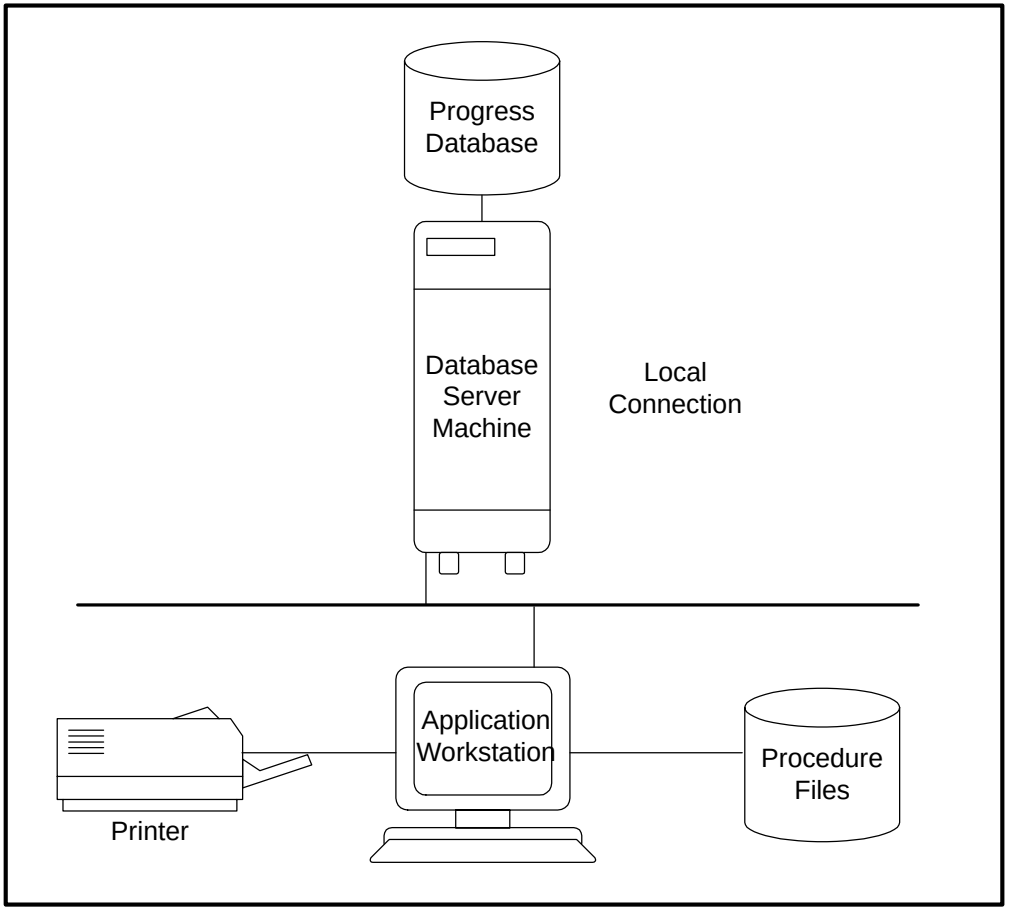


Figure 7–6: Simple Progress Network Configuration

This configuration is a TCP/IP network without file servers. There are no shared resources except the Progress database. There is a hard disk on both the workstation and database server machine. A printer is also attached to the workstation. Progress is installed on each node.

A workstation in this configuration often supports multiple users and clients (for example, a system with multiple terminals) that share the local printer and Progress application. The database server machine is usually a high-performance back-end processor that can also support local self-service clients. This network configuration, with the Progress database local to the database server machine, ensures full database integrity. With all files stored local to each node, it generally (but not always) provides the highest performance on a LAN.

Figure 7–7 shows a dedicated network file server, dedicated Progress database server machine, and application workstations. Although the figure shows a limited number of workstations, file servers, and database server machines, there can be more of each.

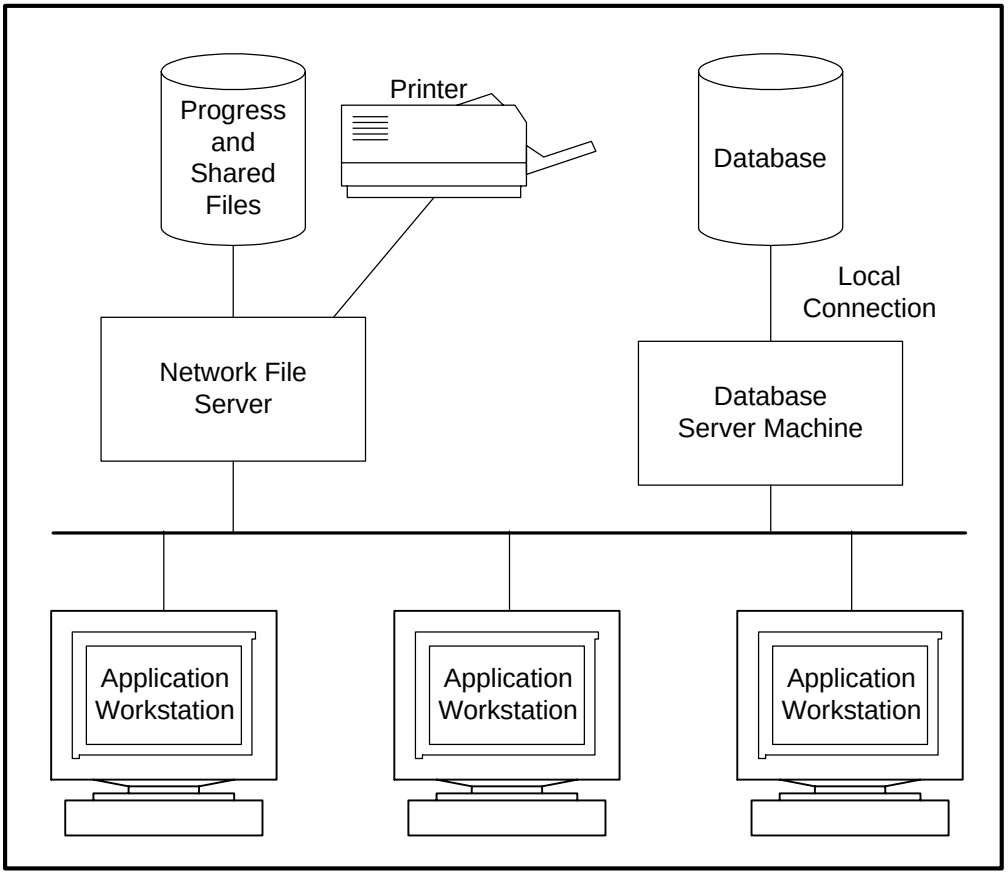


Figure 7–7: Network File Server For Application Files

This is a configuration typical for LANs with file servers and network operating systems. A hard disk and a printer are attached to the network file server, and an additional hard disk is attached to the Progress database server machine. The Progress database is on the disk drive that is locally attached to the Progress database server machine. Progress and all application procedures are installed on the file server and shared by all other nodes.

This network configuration ensures full database integrity and high performance, limited only by network and application performance capabilities.

Figure 7–8 shows a network file server doubling as a Progress database server machine and disk-optional application workstations. Although the figure shows a limited number of workstations, file servers, and database server machines, there can be more of each.

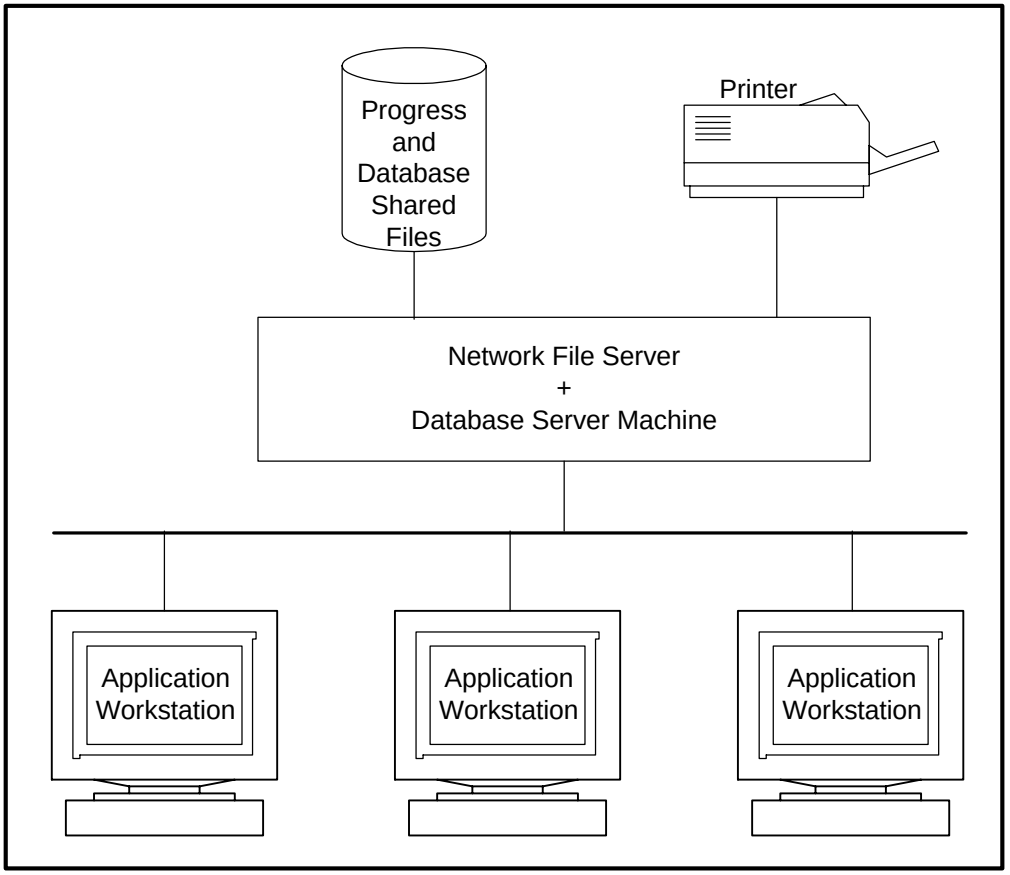


Figure 7–8: Network File Server As a Database Server

This is a configuration you might find on a LAN with a powerful file server running a multi-tasking operating system. Progress, application procedures, and the Progress database are all installed on the file server and are shared by the other nodes.

This network configuration provides full database integrity and acceptable performance on a file server with high-speed CPU and I/O resources.

NOTE: Avoid doubling a network file server as a database server machine on low-capacity nodes or on nodes where the database server machine can run only in an emulated environment.

Figure 7–9 shows database files residing on a network file server.

CAUTION: On Linux systems, Progress Software recommends you store the database on a disk that is locally connected to the database server. Do not mount the database files on a network file system.

This network configuration runs the risk of compromising database integrity if the network file server or database server machine crashes, because the before-image (BI) file must be on the network file server, making synchronous writes to it impossible. Performance also depends on whether network file server I/O efficiency compensates for traffic across the network.

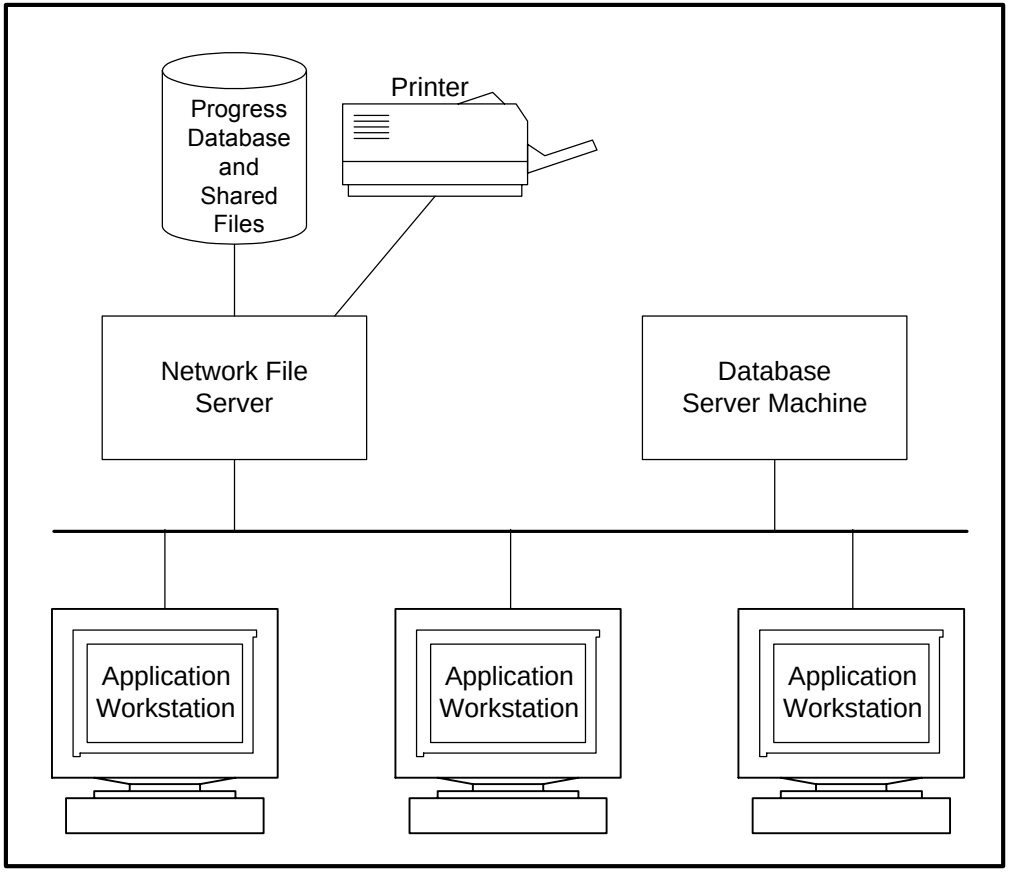


Figure 7–9: Network File Server For Application and Database Files

Figure 7–10 shows the simplest LAN configuration with the Progress AppServer. The Progress AppServer enables you to distribute portions of your business application throughout your enterprise. The flexible, modular use of computer resources provides greater performance.

An application server running on the application server machine connects through shared memory to a Progress database and has access to a set of procedure files. A Progress 4GL application runs at the application workstation, connects to the application server running on the Progress AppServer machine, and sends the requests to the application server to run remote procedures. The procedure execution and database access occur in a remote Progress session context.

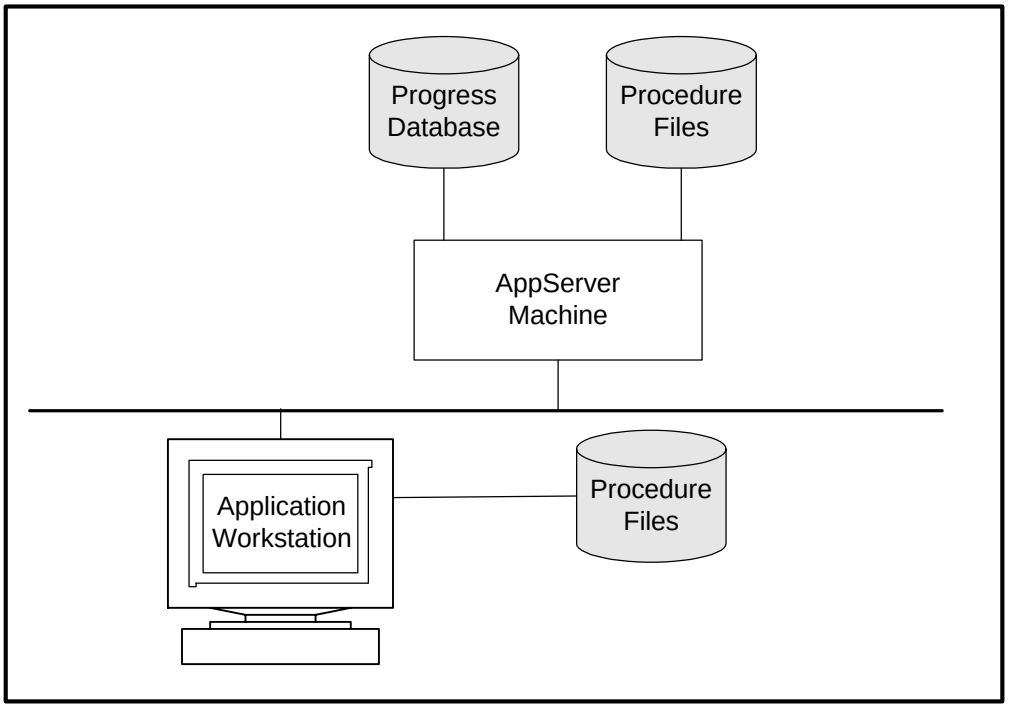


Figure 7–10: LAN Configuration With the Progress AppServer

In more complex implementations of the Progress AppServer, an application server can connect to another application server in order to connect with a database. For more information about the Progress AppServer, see *Building Distributed Applications Using the Progress AppServer*.

7.6 Preparing To Run Progress In a TCP/IP Network

You can make Progress operational in a network environment by following these guidelines:

- Identify and configure the nodes on your network for use as application workstations, database server machines, application server machines, and network file servers.
- Install Progress on each node, or if your network has a network file server, install Progress on the file server. For more information, see Chapter 2, "Installing Progress."
- If any application workstations and database server machines have incompatible processors or operating systems, you must install the appropriate Progress product on each node.
- Set up network system files on each node.
- If you are using a network file server, make its resources, including printers and directories, available to all other nodes that require them.
- If you installed Progress on a network file server, you might want to distribute the appropriate Progress system files to the compatible application workstations and database server machines that use them. This takes advantage of networks where the local file and data access is faster than using the network.
- Set up your Progress databases on each file server, database server, and application server machine.

7.6.1 Configuring UNIX Machines In a TCP/IP Network

Each UNIX machine in a TCP/IP network has three components:

- A file in `/etc/hosts` (or elsewhere) or a Network Information System that lists machine names and network addresses. At a minimum, the file must be in the machines where each Progress server and each application server (Progress AppServer environment) is running.
- A file in `/etc/services` (or elsewhere) or a Network Information System that assigns a name, port number, and communication protocol to the server process for each Progress database or to the application server in a Progress AppServer environment. This is a sample entry in the services file:

```
sportssv 2500/tcp
```

NOTE: The main database server process in called the *database broker* on systems running shared-memory versions of Progress. For more information on shared-memory systems, see the "Shared-memory Configurations" section.

- A file in `/etc/protocols` that identifies the protocols available to the network in the TCP/IP protocol suite.

7.6.2 Installing Progress On Your TCP/IP Network

When installing Progress on your network, keep these two basic considerations in mind:

- Where to place your database
- Where to place your Progress executables and r-code files

Locating Your Database

Place your database on the hard disk of the machine that runs the Progress server. If you place the database on a remote file server, synchronous writes are lost along with your database's integrity, in the event of a system crash.

Synchronous writes ensure database integrity by flushing system buffers directly to disk. This is especially important for maintaining the before-image (BI) file. Therefore, if you must keep your database separate from the database server machine, use the Before-image Filename (-g) startup parameter to keep the before-image file local to the database server.

NOTE: Remote Progress clients do not have to be concerned about synchronous writes because they do not write to the database.

7.6.3 Typical TCP/IP Configuration With a Hard Disk On Each Machine

Figure 7–11 shows the configuration for a typical network when there is a hard disk on each machine and no file server is used.

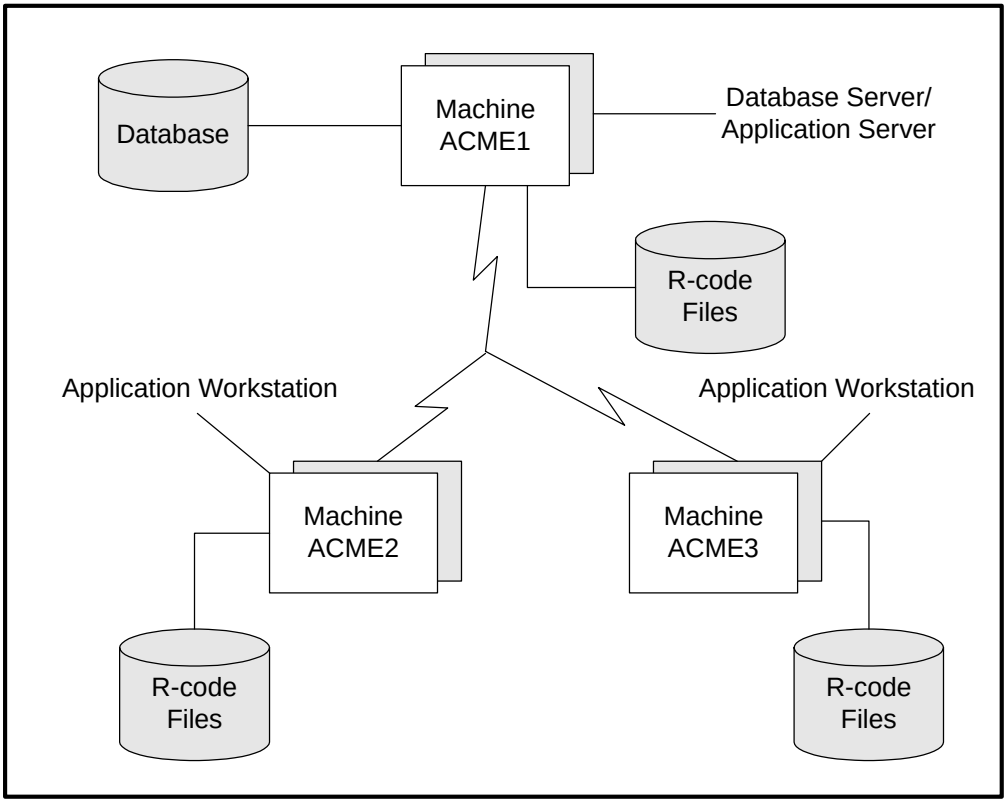


Figure 7–11: Typical TCP/IP Configuration (File Server Not Used)

When you use this configuration, you must install Progress on each machine in the network. In Figure 7–11, the client machines do not have to be running the same operating system.

7.6.4 Setting Up Network Files To Run Progress

There are several files you must check, and modify if necessary, before you can run Progress on your network. The filenames and locations might differ for different operating systems and TCP/IP implementations, but the functional contents are identical. Table 7–4 lists these files.

Table 7–4: TCP/IP Network Files

File	Purpose
`hosts`	Lists machine names/network addresses.
`services`	Lists Progress server/port number.
`protocols`	Defines system protocols.

7.6.5 Configuring Progress On a Network Operating System

This section describes preparations that you can make to promote efficient and reliable Progress operation in a network operating system (NOS) environment, that is, a network environment that includes one or more network file servers that provide a common set of resource sharing and security services to other nodes. This section describes some of the more general considerations. For information on specific requirements and configurations on your particular network type, see any additional sections in this chapter that describe your network type.

Making Network Resources Available

Once you have installed Progress, you must make sure that each of your application workstations and Progress AppServer machines has access to Progress system files, application files, and any other necessary network resources (such as printers). Each NOS provides a set of commands or utilities to make these resources available across the network. In general, you set up pointers to remote resources so that each workstation can access them as though they were local to the workstation. These pointers can be in the form of logical drives (for example, on Windows nodes) or mounted directory paths (for example, on UNIX nodes).

For example, suppose that you have a network configuration similar to the one shown in Figure 7–12, with Progress installed on the network file server.

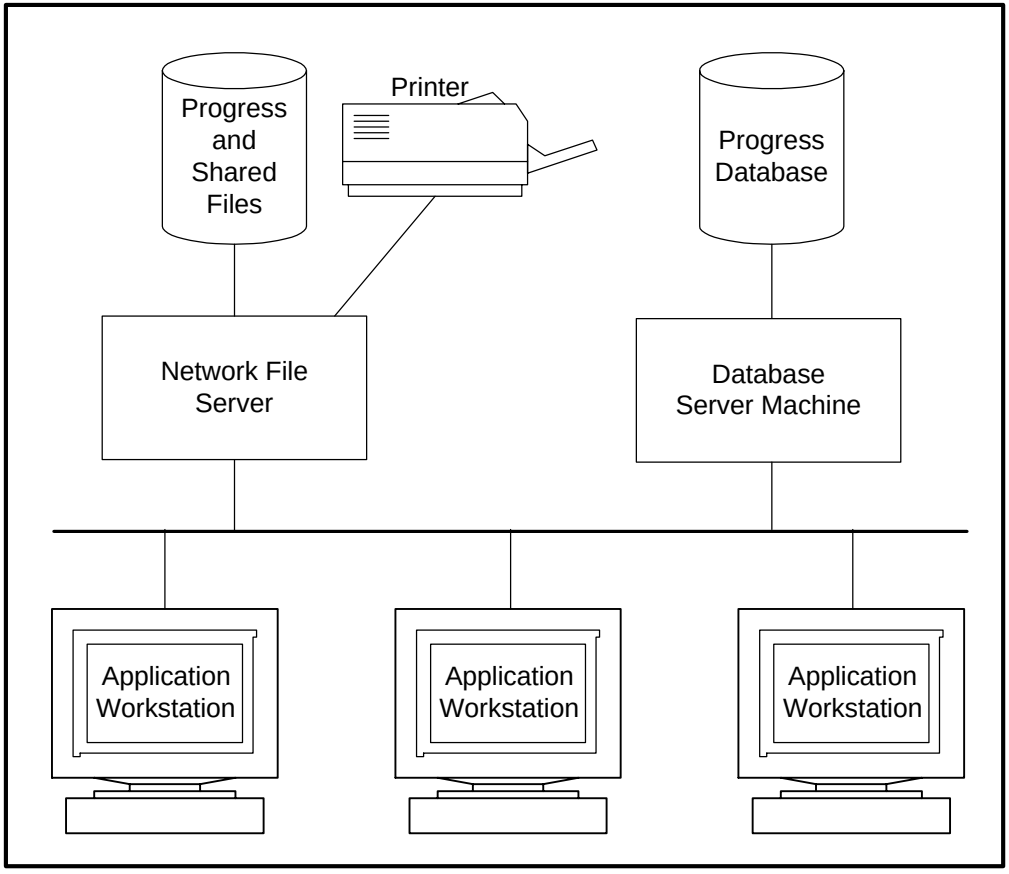


Figure 7–12: Sample Network Resources

For more information on making network resources available, see the documentation for your network and operating system.

Setting Network Resource Attributes

After you have made network resources available, you must make sure that they possess the necessary attributes to allow all application workstations to access them simultaneously. Each NOS provides a different means of setting the attributes to make network resources shareable.

For example, suppose you have Progress installed on a network file server and you want to grant all workstations access to Progress. You must set the correct resource attributes on the Progress installation directory. The commands used to set resource attributes vary from network to network. For further information on how to use these or equivalent commands for your network, see the documentation for your network and operating system.

Granting User Access Rights

After making Progress network resources available and setting resource attributes, you might have to grant access rights to client users and Progress AppServer machines in the network. Depending on your network, these access rights can include attributes such as read, execute, or open permissions that you must set for each user. See the network documentation for details about how to grant user access rights.

NOTE: User rights in a Progress AppServer configuration are assigned to the machine where the application server resides, not to the user's client machine.

Remember that a Progress database server can be a user on your network. Like application workstations, it might need user access rights granted to it. If you locate any database files on your network file server, be sure to grant the Progress database server the necessary rights to access the network directory that contains the database.

8

Configuring Progress Unified Broker Products

This chapter describes the framework and tasks for configuring the common components of Progress Unified Broker products that run on UNIX. For more information on product-specific configurations, see the documentation for each product as explained in this chapter. These products all support multi-tier or client/server configurations using the Progress NameServer and include the Progress AppServer, WebSpeed, SonicMQ Adapter, and, when configured to use the Progress NameServer, the Progress DataServers for ODBC (Windows only), Microsoft SQL Server, and ORACLE.

Specifically, this chapter contains the following sections:

- The Progress Unified Broker Administration Framework
- Configuring Progress Unified Broker Components
- Setting Up the Progress Unified Broker Environment
- Starting Progress Unified Brokers
- Understanding Load Balancing
- Understanding Server-level and Connection-level Fault Tolerance

8.1 The Progress Unified Broker Administration Framework

Progress integrates all management functions for Unified Brokers in a single administration framework. Figure 8–1 shows an overview of this framework.

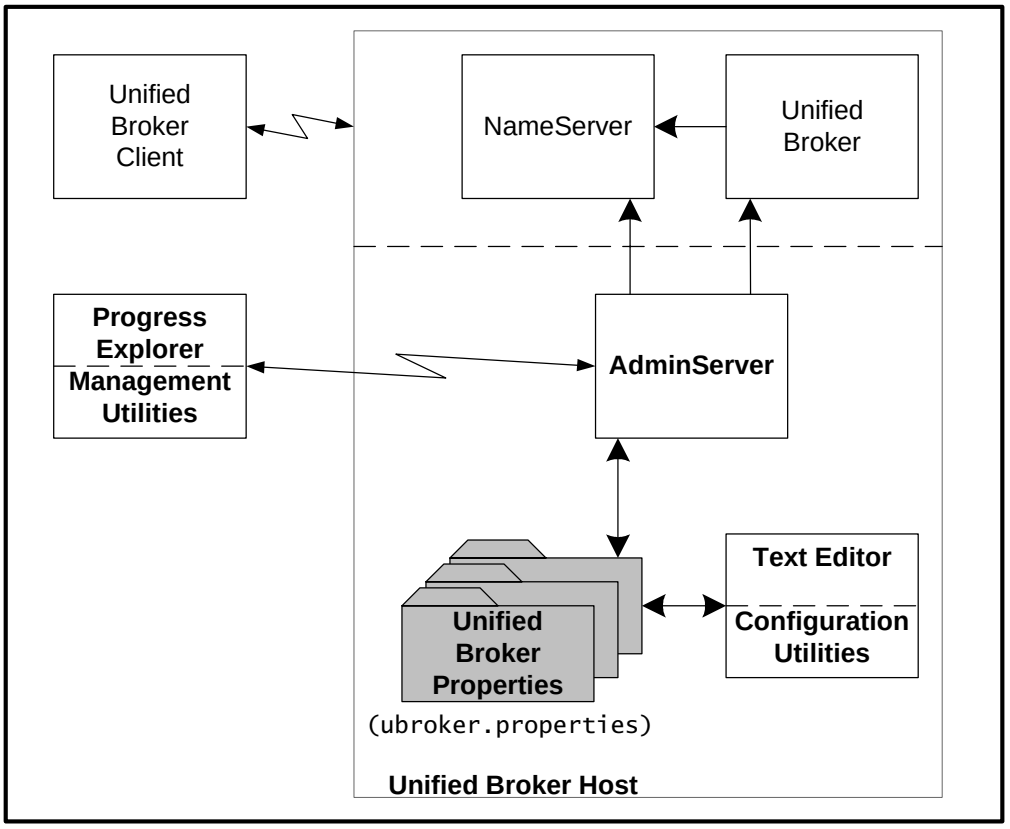


Figure 8–1: Progress Unified Broker Administration Framework

Figure 8–1 shows all the Unified Broker components running on the same machine. You can install some of these components on separate machines connected on the same network. In that case, a separate AdminServer and `ubroker.properties` file exists on each machine for access by the Progress Explorer. For more information, see the "Distributing Machines For a Unified Broker Product" section.

The core of this framework is a common process, the AdminServer process, for managing the AppServer, WebSpeed, SonicMQ Adapter, and DataServer products. The AdminServer supports the Progress Explorer and management utilities to provide similar configuration and management functions for all of these products.

Complete administration for a Unified Broker application potentially involves all of the components in Figure 8–1. However, for some components and administration tasks you can use AdminServer-based management utilities (including the Progress Explorer) or a text editor and configuration validation utilities to accomplish the task; for others you can only use the AdminServer-based utilities. Table 8–1 summarizes the components and possible administration approaches.

Table 8–1: Unified Broker Administration Summary

Components	Configuration	Run-time Management
Unified Broker clients	AdminServer-based utilities or text editor and configuration validation utilities (WebSpeed Messenger only). Otherwise, as specified for your client product.	AdminServer-based utilities (WebSpeed Messenger only). Otherwise, as specified for your client product.
NameServers	AdminServer-based utilities or text editor and configuration validation utilities.	AdminServer-based utilities.
Unified Brokers	AdminServer-based utilities or text editor and configuration validation utilities.	AdminServer-based utilities.
Unified Broker properties file, `ubroker.properties`	AdminServer-based utilities or text editor and configuration validation utilities.	NA

NOTE: The NameServer component provides location transparency and load balancing capabilities to these servers that can be configured to use it: AppServer, WebSpeed Transaction Server, Sonic Adapter, and DataServer products. Also, all communications with the NameServer use UDP. The default configuration is to use the NameServer. However, it is an option feature. If you elect not to use NameServer for any of these servers, then these features are not available.

This chapter provides specific information on the NameServer and outlines the general procedures for administration of the remaining components. For specific information on administration of all application components for a Unified Broker product, see your product documentation:

- **AppServer** — *Building Distributed Applications Using the Progress AppServer*
- **AppServer Internet Adapter** — *Progress on the Web*
- **DataServers** — Depending on your product:
  - *Progress DataServer for Microsoft SQL Server Guide*
  - *Progress DataServer for ODBC Guide*
  - *Progress DataServer for ORACLE Guide*
- **SonicMQ Adapter** — *Progress on the Web*
- **WebSpeed** — *WebSpeed Installation and Configuration Guide*

8.1.1 Unified Broker Clients

The clients of Unified Broker products include:

- **AppServer** — 4GL clients (including other AppServers and WebSpeed instances) and Open Clients
- **AppServer Internet Adapter** — 4GL clients (including AppServer and WebSpeed instances)
- **DataServers** — 4GL clients (including AppServer and WebSpeed instances)
- **SonicMQ Adapter** — 4GL clients (including AppServer and WebSpeed instances)
- **WebSpeed** — The WebSpeed Messenger, which directs Web client requests to WebSpeed Transaction Servers

Of these clients, you can use the Unified Broker administration framework to manage only WebSpeed Messengers. For specific information on configuring these clients, see your Unified Broker product documentation.

8.1.2 Progress Explorer and Management Utilities

Progress provides two ways to manage local or remote Unified Brokers:

- Progress Explorer tool
- Command-line management utilities

Progress Explorer Tool

The Progress Explorer is a graphical user interface that provides an easy way for you to manage Progress servers. The Progress Explorer tool runs as a Windows client of the Progress AdminServer to manage the following Progress servers on UNIX:

- AppServer
- AppServer Internet Adapter
- Database
- DataServer for Microsoft SQL Server
- DataServer for ODBC (Windows only)
- DataServer for ORACLE
- NameServer
- SonicMQ Adapter
- WebSpeed Messenger
- WebSpeed Transaction Server

Using the Progress Explorer tool, you can remotely perform the following tasks on UNIX-resident Progress servers:

- Create new instances of Progress servers and configure their property settings
- Remove existing Progress server instances
- Modify property settings of existing Progress server instances
- Start, stop, manage, monitor the status of Progress server execution

This chapter describes basic Progress Explorer administration for all of these server products except the Database. For information on Progress Database administration, see the *Progress Database Administration Guide and Reference*. For more information on working with the Progress Explorer tool to manage all the Progress servers, see the *Progress Installation and Configuration Guide Version 9 for Windows* and the Progress Explorer online help.

Command-line Management Utilities

The command-line management utilities are a set of utilities for both Windows and UNIX that allow you to manage existing Unified Broker configurations. On UNIX, these include the:

- **ADAPTMAN utility** — To manage SonicMQ Adapters
- **AIAMAN utility** — To manage AppServer Internet Adapters
- **ASBMAN utility** — To manage AppServers
- **NSMAN utility** — To manage NameServers
- **ODBMAN utility** — To manage DataServers for ODBC (on Windows only)
- **ORAMAN utility** — To manage DataServers for ORACLE
- **WTBMAN utility** — To manage WebSpeed Transaction Servers

Like the Progress Explorer tool, the command-line management utilities run as clients of the Progress AdminServer to manage the NameServer and Unified Broker products. This chapter describes how to manage the NameServer using the NSMAN utility and introduces how to manage the Progress Unified Brokers using their own command-line management utilities.

Using these utilities, you can locally or remotely start, stop, manage, and monitor the status of Unified Broker execution. Unlike the Progress Explorer, they do not help you create, remove, or modify properties for Unified Broker configurations.

For more information on starting and managing the NameServer using the NSMAN utility, see the "Starting Progress Unified Brokers" section. For more information on starting and managing the NameServer using the Progress Explorer from a Windows machine, see the chapter on configuring Progress Unified Broker products in the *Progress Installation and Configuration Guide Version 9 for Windows*. For more information on using the Progress Explorer and management utilities for your Unified Broker product, see your product documentation.

The rest of this chapter describes how to configure NameServers and begin configuration and operation of your Unified Broker product.

8.1.3 NameServers

A *NameServer* is a single process that mediates client connections for a set of Unified Brokers that have registered with it. Any number and type of Unified Broker instance can register with a single NameServer, and each Unified Broker instance can register with exactly one NameServer. The NameServer that a broker instance registers with is the broker's *controlling NameServer*.

NOTE: The NameServer is an optional component that you can use to support these servers: Progress AppServer, SonicMQ Adapter, WebSpeed, and DataServer products. However, since Progress provides the NameServer by default, the information and graphics throughout this chapter assume you are using the NameServer. For specific information about the optional usage of the NameServer, see your product documentation.

When a Unified Broker instance starts up, it registers with its controlling NameServer by sending its location and other configuration information to help resolve a connection request for the broker. Part of this registration information is the Application Service that the Unified Broker supports. An *Application Service* is a designation for the particular business function that a Unified Broker provides. For more information on Application Services, see the "Unified Broker" section.

NameServers provide the following services for a Unified Broker product:

- **Location Transparency** — A requesting client does not need to know the network location of a Unified Broker instance. When a client attempts to create a connection to a Unified Broker instance, it first requests the connection from a NameServer to a broker that provides a specified Application Service. The NameServer then locates and assigns a broker to complete the connection that provides the specified Application Service.

- **Server-level Fault Tolerance and Load Balancing** — If you have installed the load-balancing option, you can provide server-level fault tolerance, where the NameServer can select from several Unified Broker instances to satisfy a client request.This option also allows you to balance connection load among multiple Unified Broker instances that provide the same Application Service. The NameServer then assigns connections among several Unified Broker instances based on a weight factor that you configure for each instance. For more information on server-level fault tolerance, see the "Understanding Server-level and Connection-level Fault Tolerance" section. For more information on how load-balancing affects Unified Broker operation, see the "Understanding Load Balancing" section.

- **Connection-level Fault Tolerance** — You can also make multiple NameServer instances available to help ensure that at least one NameServer is available even if another fails. In this type of configuration, one of several possible NameServers resolves the connection request. Thus, you can provide connection-level fault tolerance for requesting clients. For more information on connection-level fault tolerance, see the "Understanding Server-level and Connection-level Fault Tolerance" section.

These NameServer services are available to the following servers: AppServer, WebSpeed Transaction Server, Sonic Adapter, and DataServer products. Also, all communications with the NameServer use User Datagram Protocol (UDP). The default configuration is to use the NameServer. However, the NameServer is an option feature. If you elect not to use NameServer for any of these servers, these features are not available.

For more information on how your Unified Broker product uses NameServers, see your product documentation.

8.1.4 Unified Broker

Unified Broker products include a Unified Broker process that is the initial point of client connection to a Unified Broker product instance. It is the process responsible for managing other process resources that are part of the product, and making those resources available to clients. As such, the Unified Broker process:

- Registers with a controlling NameServer the following information:
  - The broker's location on the network
  - The weight factor that you specify for load balancing
  - The Application Service that you specify.
- Manages connections between clients and the Unified Broker instance.
- Provides other services unique to the Unified Broker product (AppServer, SonicMQ Adapter, WebSpeed, or DataServer). For example, it maintains the status of each 4GL process running on an AppServer and scales the number of processes according to changing demand.

Unified Broker Types

One Unified Broker process manages a single Unified Broker product instance of the same type (AppServer, SonicMQ Adapter, WebSpeed, or DataServer). The product documentation for each Unified Broker type identifies this Unified Broker process by a different name, as follows:

- **AppServer** — Application Broker
- **DataServers** — DataServer Broker
- **SonicMQ Adapter** — SonicMQ Adapter Broker
- **WebSpeed** — WebSpeed Transaction Broker

For more information on how to configure the Unified Broker process for a product, how clients specify connections, and how the Unified Broker manages connections with clients, see your product documentation.

Application Services

The Application Service that a Unified Broker provides is identified by a list of one or more names that you can optionally specify during broker configuration. Each Application Service name you specify is an arbitrary designation for the business function that the Unified Broker instance provides.

The NameServer maintains a separate Application Service name space for each Unified Broker type. So, an AppServer, WebSpeed Transaction Server, and DataServer instance can each register the same Application Service name with the same controlling NameServer without conflict. However, with the load-balancing option, if you have multiple Unified Broker instances of the same type register the same Application Service name with the same controlling NameServer, you must guarantee that each Unified Broker instance provides exactly the same functionality. For AppServers and WebSpeed Transaction Servers, this means providing the same application procedures and database resources for all instances. For DataServers, this means accessing the same database for all instances.

If, for example, you use the same Application Service name to identify functionality on several AppServers, each of which support different remote procedures and database connections, multiple requests from the same client application are likely to provide inconsistent results.

The Default Service

You actually do not have to designate an explicit Application Service for a Unified Broker. Instead, you can specify that the broker supports the *default service*. The default service is a special Application Service designation that supports default client connection requests. Thus, any Unified Broker that supports the default service and is of the appropriate type (AppServer, WebSpeed, or DataServer) can satisfy a connection request from a client that does not specify an Application Service name as part of its connection request.

8.1.5 Unified Broker Properties File (ubroker.properties)

Progress stores the configurations for all the component configurations managed by the Unified Broker administration framework in a properties file, `ubroker.properties`. The UNIX and NT `ubroker.properties` files are the same except for platform-specific differences (for example, differences in directory path separators and the differences between environment variable references in UNIX and registry references in NT).

On UNIX, this file includes the configurations for:

- AppServers
- AppServer Internet Adapters
- NameServers
- Oracle DataServers
- SonicMQ Adapters
- WebSpeed Transaction Servers

There is one copy of this file local to each Progress installation. The AdminServer reads and updates the file according to your instructions using the Progress Explorer and management utilities. The `ubroker.properties` file is installed in the `properties` subdirectory of the Progress installation directory (for example, `$DLC/properties/ubroker.properties` on UNIX or `%DLC%\properties\ubroker.properties` on NT). In order for the AdminServer to access the properties file, the file must reside in this directory.

NOTE: When you install a new Unified Broker version, either to overwrite an existing installation or to add additional Progress components to the current installation, the installation program automatically renames the existing `ubroker.properties` file, located in the `properties` directory, to `ubroker.properties-sav`.

On UNIX, you have to modify this file using a text editor. For more information, see the "Using a Text Editor and Configuration Utilities" section. For more information on editing the properties file to configure NameServers, see the "Editing the Properties File" section. For more information on managing the `ubroker.properties` file for your Unified Broker product, see your product documentation.

8.1.6 Using a Text Editor and Configuration Utilities

You can update the `ubroker.properties` file manually by opening it locally on the machine where it resides using any text editor. This is the only method available to update Unified Broker configurations locally on a UNIX platform.

Guidelines For Editing the Properties File

In general, it is recommended that you update all UNIX configurations remotely using the Progress Explorer tool. If you must update a UNIX configuration locally using a text editor:

- You should not directly change the values in the `ubroker.properties` file unless you have a complete understanding of how the changes affect Unified Broker components. When possible, always use the Progress Explorer to make all changes to this file.
- Always make a copy of this file, edit the copy, and verify the result before replacing the original with your edited copy.
- For complete definitions of all the properties and detailed information on how to set them, see the comments included in the properties file.
- Verify the result using the appropriate configuration validation utilities.

The file consists of a hierarchical structure of configuration entities, where parent entities provide configuration information that you can override or extend in each child entity. Each configuration entity has a name that begins the entity definition, and the definition contains configuration settings for one or more products or product instances.

Validating the Properties File

Progress provides a set of configuration validation utilities that you can use to verify the correctness of any changes that you make to this file.

NOTE: If you always use the Progress Explorer from a remote Windows machine, you never have to use these utilities.

On UNIX, you can use the:

- **ADAPTCONFIG utility** — To validate SonicMQ Adapter configurations
- **AIACONFIG utility** — To validate AppServer Internet Adapter configurations
- **ASCONFIG utility** — To validate AppServer configurations
- **NSCONFIG utility** — To validate NameServer configurations
- **ORACONFIG utility** — To validate DataServer for ORACLE configurations
- **WSCONFIG utility** — To validate WebSpeed configurations

When you validate the `ubroker.properties` file, make sure that you run the configuration utilities that apply to all the component configurations stored in the file.

For more information on editing and validating the `ubroker.properties` file to configure a NameServer, see the "Editing the Properties File" section. For more information on editing and validating the file for each Unified Broker product, see your product documentation.

8.1.7 Overview Of Unified Broker Administration

There are two preliminary tasks you must complete before you can begin configuring and operating a Unified Broker product:

- Configure and set up all machines involved in product installation and operation. This depends on how you plan to distribute your product and its applications on a network. For more information on configuring Progress products on a network, see Chapter 7, "Configuring Progress."
- Install the necessary product components. Typically, this involves installing, on one or more network machines, the Progress Unified Broker product and whatever software components are required to use the product, such as Progress client or Web server software. If you plan to configure fault-tolerant servers or use load balancing, you must install a product that includes load balancing or install the load-balancing option for your product. For more information on Progress product installation, see the relevant chapters of this manual or the *Progress Installation and Configuration Guide Version 9 for Windows*.

For more information on the distributing resources in a Unified Broker environment, see the "Distributing Machines For a Unified Broker Product" section in this chapter.

Once you complete these preliminary tasks, you can configure and start up Unified Broker instances, as shown in Figure 8–2.

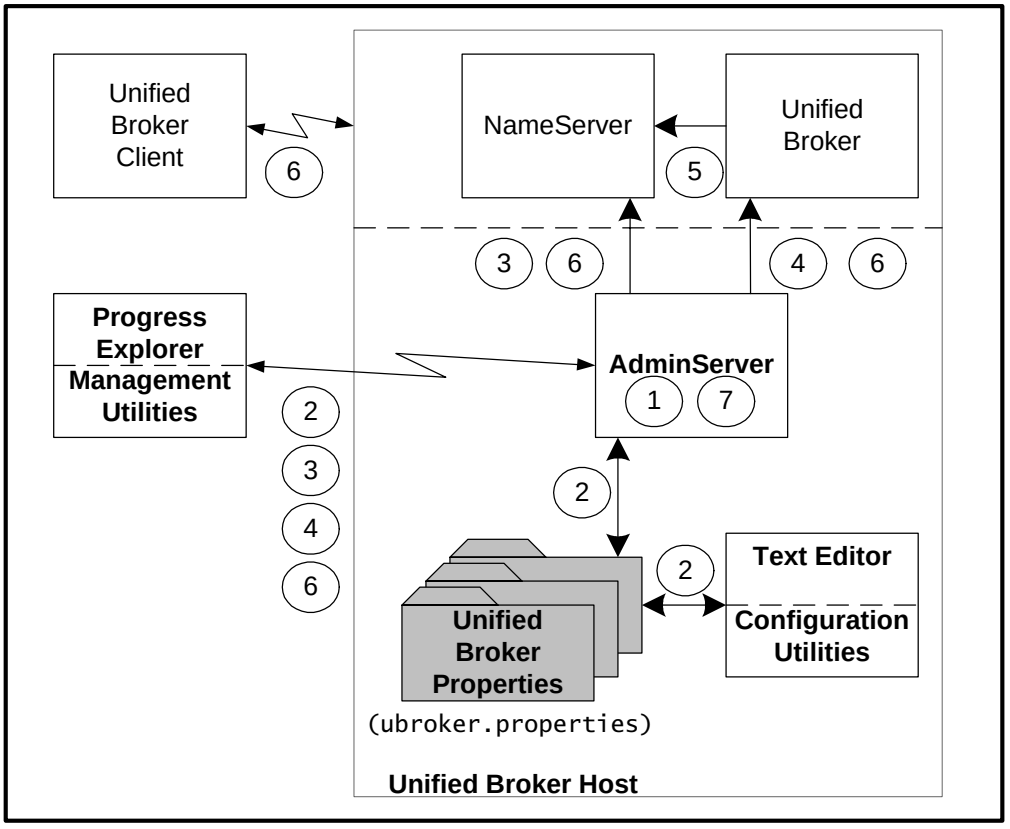


Figure 8–2: Progress Unified Broker Administration Overview

These are the steps, shown as circled numbers, in Figure 8–2:

1 ♦ Start the AdminServer process on each Unified Broker and NameServer installation machine. On NT, Progress installs the AdminServer as an NT service that starts automatically at system boot time. On UNIX, you can have the AdminServer started at system startup by editing your boot script to execute the PROADSV command. For more information on starting the AdminServer, see Chapter 5, "Setting Up Your Environment."

2 ♦ Once the AdminServer is running, use the Progress Explorer to create and modify NameServer and Unified Broker configurations. You can use the Progress Explorer locally on an NT machine or remotely from any Windows machine to access configurations installed on UNIX or NT machines. You can also use a text editor to create these configurations directly in the `ubroker.properties` file that is local to the NameServer or Unified Broker product installation machine, whether or not the AdminServer is running. For more information on configuring NameServer and Unified Broker components, see the "Configuring Progress Unified Broker Components" section.

3 ♦ Using the Progress Explorer (or the NSMAN utility), start the NameServer that you have configured to coordinate client access to one or more Unified Broker instances. For more information, see the "Starting Progress Unified Brokers" section.

NOTE: Figure 8–2 shows the NameServer running on the Unified Broker installation machine. However, the NameServer can run on any network machine that is accessible to all of the clients and Unified Broker instances that it coordinates.

4 ♦ Using the Progress Explorer (or the management utility for your Unified Broker product), start up each Unified Broker instance coordinated by the NameServer that you started in Step 3. As it starts, each Unified Broker instance starts additional processes or accesses resources, depending on the product and its configuration.

5 ♦ After a Unified Broker starts up, it registers its network address and the Application Services it supports with the NameServer that you started in Step 3.

6 ♦ A client can now make a Unified Broker connection request after you ensure that it knows:

- The correct network location of the NameServer to access
- The Application Service name required to connect to the broker that the client needs

At any time after Step 5, you can also use the Progress Explorer or the appropriate management utility to shut down or query the status of any running NameServer or Unified Broker instance.

7 ♦ You can shut down an AdminServer process at any time. If you have not already shut down the NameServer and the Unified Broker instance that it controls, the instance shuts down automatically when you shut down the AdminServer.

During Unified Broker operation, in addition to checking NameServer and Unified Broker status using the Progress Explorer and utilities, you can also review log files being generated by the NameServer and Application Serve instance. For more information on working with these files, see the "Configuring Progress Unified Broker Log Files" section.

8.1.8 Distributing Machines For a Unified Broker Product

Figure 8–3 shows the maximum distribution of resources that you can achieve for any single Unified Broker product instance. For WebSpeed, the Unified Broker Client (WebSpeed Messenger) also resides on the same machine as the Web server, and Web clients (browsers, not shown) can reside anywhere on the Internet, intranet, or extranet serviced by the Web server.

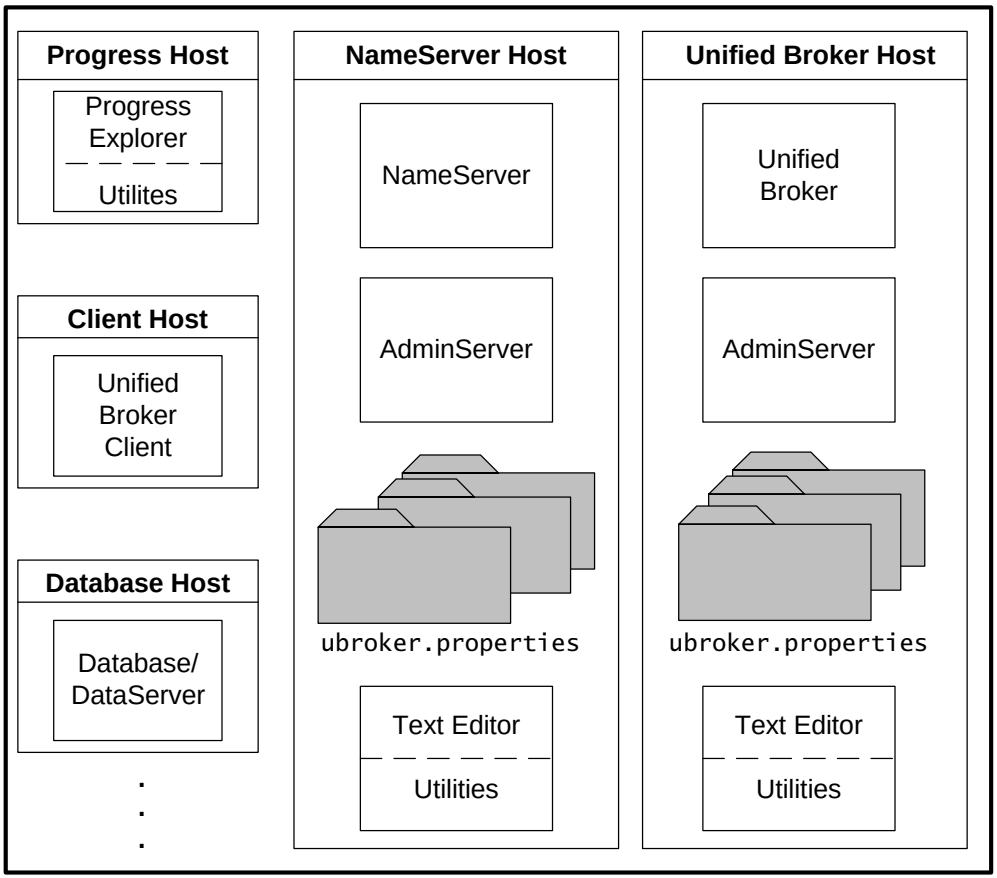


Figure 8–3: Maximum Unified Broker Machine Distribution

The separate database host for a DataServer applies only to WebSpeed or the AppServer. For a DataServer, the Unified Broker host is the DataServer host. Note that whether the target database management system (DBMS) can reside on a separate database host or must reside on the same machine as the DataServer host depends on the DataServer and its platform.

Thus, you can distribute a Unified Broker instance and its management components among three separate machines on the same network. Also, the Unified Broker client and any single Progress data source can reside on separate machines, for a total distribution of five machines plus any machines in the same network that you might require for additional databases, DataServers, or other Unified Broker instances.

Note that for all Unified Broker products, these components must reside on the same machine:

- Unified Broker instance and associated processes
- AdminServer
- Unified Broker properties file (`ubroker.properties`)

Similarly, the NameServer, if it resides on its own machine, is installed with an administration framework, including an AdminServer and its own `ubroker.properties` file. If you use a text editor to modify the `ubroker.properties` file, the editor and configuration utilities must reside on the same machine as the Unified Broker or NameServer instance, or have network file system access to the respective Unified Broker and NameServer installation files.

8.1.9 Configuring NameServer Communications

Both clients and Unified Brokers use UDP to communicate with NameServers. User Datagram Protocol (UDP) is an Internet standard, combined network layer, transport layer, and session layer protocol that provides a mechanism for connectionless communications. The connectionless nature of this protocol affords built-in benefits, such as the ability to implement fault-tolerant NameServers for client connections. For information on connection-level fault tolerance, see the "Understanding Server-level and Connection-level Fault Tolerance" section.

NOTE: The NameServer requires UDP communications with clients and Unified Brokers. If you do not want to use UDP, you can elect not to use the NameServer component.

To establish a Unified Broker connection, a Unified Broker client specifies the location of the NameServer that provides the connection. To register with a NameServer, a Unified Broker must specify the location of the NameServer where it needs to register. To specify the NameServer location, both components must know the UDP port number (or service name) on which the NameServer is listening and host address of the machine where it resides.

Specifying NameServer Ports and Hosts

4GL clients provide this information when they specify the CONNECT( ) method, and AppServer Open Clients provide it using an equivalent Open Client method. You must specify this information for all other Unified Broker components in the Progress Explorer or directly in the Unified Broker properties file (`ubroker.properties`). For more information, see the "Configuring Progress Unified Broker Components" section.

If the component uses a UDP service name rather than a port number, you also must ensure that the `services` file on the component host (or Network Information Services (NIS), if used) properly defines the UDP service name.

Editing the services File

The `services` file stores the service name, port number, and protocol for various services on the network. For each NameServer that the component accesses, the `services` file on the component host must specify a service name associated with the NameServer UDP port number (default, 5162). Thus, if the component connects using the service name `namesv`, you might enter the service name definition for the `services` file extract shown in Figure 8–4.

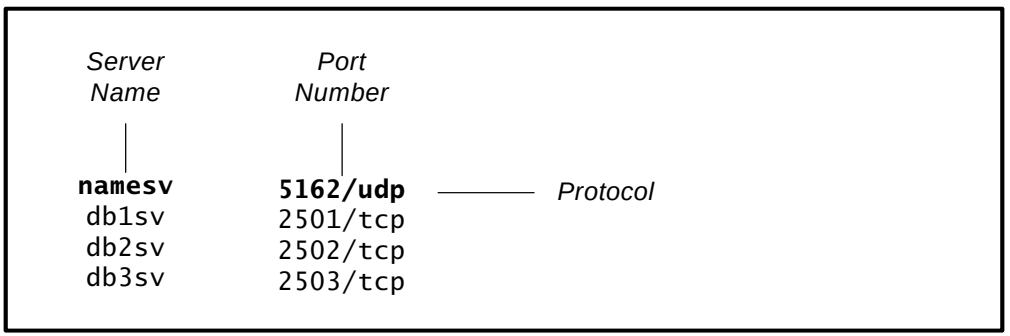


Figure 8–4: Sample Unified Broker Client Services File

8.2 Configuring Progress Unified Broker Components

You can use the Progress Explorer to configure NameServer and Unified Broker instances, locally on NT or remotely for both NT and UNIX hosts. If you plan to configure instances directly on a UNIX host, you must edit the properties file (`ubroker.properties`) for each NameServer and Unified Broker instance directly on the host. The properties file that comes installed with your Unified Broker product includes one sample NameServer and Unified Broker instance for each type of Unified Broker that you can use as a guide.

8.2.1 Downloading NameServer Executables

If you need a NameServer executable, for example, for a different deployment platform, you can download it from the Progress Web site at www.progress.com:

1 ♦ Choose the Tech Support page.

2 ♦ Choose the Product Updates and Documentation page.

3 ♦ Go to the Downloads section, and follow the instructions to download the Progress NameServer executable for your Progress version and platform.

8.2.2 Order Of Configuration

To configure a Unified Broker instance, you generally configure components in this order:

1. Controlling NameServer and any replicated or neighbor NameServers

2. Unified Broker product components

In the Progress Explorer, you must have an initial configuration for the controlling NameServer instance to identify it when you configure your Unified Broker product instance. Editing the properties file, you can configure these components in any order. Whatever order you configure these components, you must have the controlling NameServer configured and running before clients can access your Unified Broker instance.

For more information on configuring Unified Broker components on UNIX from a remote Windows machine using the Progress Explorer, see the *Progress Installation and Configuration Guide Version 9 for Windows* and the Progress Explorer online help. The rest of this section describes how to configure Unified Broker components using a text editor and configuration utilities that run on UNIX.

8.2.3 Configuring and Using NameServer Instances

You can configure two types of NameServer instance, determined by its function as the controlling NameServer for a Unified Broker instance:

- **Local** — An instance that runs locally on the host where it is defined.

- **Remote** — An instance that references a NameServer defined and running locally on a machine that is remote from the host where the remote instance is defined.

When you configure a local NameServer instance, you can set all properties for the NameServer. When you configure a remote NameServer instance, you can only set its location (host and port) properties to identify the local NameServer instance that it references. When you want to start, stop, or obtain status on a running NameServer, you must always perform these actions on a local instance. You cannot start, stop, or obtain status on a remote NameServer instance.

How Unified Brokers Use NameServer Instances

To use a local NameServer instance as its controlling NameServer, a Unified Broker instance must run on the same machine where the local NameServer instance runs. Remote NameServer instances provide a way of having multiple Unified Broker instances use a controlling NameServer that runs on a different machine from the Unified Broker instances.

Whether local or remote, the NameServer instance that you define as the controlling NameServer must be defined on the same machine as the Unified Broker instance it controls. If the controlling NameServer instance is local, it runs on the same machine as the Unified Broker. If the controlling NameServer instance is remote, it references a NameServer running locally on a machine that is remote from the Unified Broker.

Thus, any remote NameServer instance you define must have a corresponding local NameServer instance defined on the machine where it runs, and you must define one such remote NameServer instance on each remote machine where a Unified Broker instance references this same corresponding local NameServer instance as its controlling NameServer.

NameServer Instances and Client Connections

Unified Broker clients do not use local and remote NameServer instances. Clients must direct all connection requests to a NameServer on the machine where it runs, that is, to a NameServer where it is defined as a local instance.

8.2.4 Editing the Properties File

The Unified Broker properties file, `ubroker.properties`, stores the configuration definitions for all instances of the NameServer, and all instances of any AppServer, AppServer Internet Adapter, SonicMQ Adapter, WebSpeed Transaction Server, and DataServer products that run on the same machine. Each configuration definition contains property settings, environment variables (to be set when the Unified Broker starts), and references to registry entries (if Windows NT) for each product instance. Because the same properties file stores configurations for several different products, some sections of the file are generic to several different product broker configurations.

There is one copy of this file for each Progress installation. Thus, if you install the controlling NameServer on a separate machine from the Unified Broker product that it controls, the NameServer and Unified Broker product each have their own `ubroker.properties` file. For more information on the Unified Broker properties file, see the "Unified Broker Properties File (ubroker.properties)" section.

You can configure Unified Broker products in two ways. You can use the Progress Explorer from a remote Windows machine. (For more information on configuring a Unified Broker instance using the Progress Explorer, see the *Progress Installation and Configuration Guide Version 9 for Windows* and the Progress Explorer online help.) Alternately, you can edit the ubroker.properties file by hand and use the command-line utilities to validate your changes.

Editing and Validating the Properties File

To edit the properties file directly, use a text editor such as `vi` or Notepad. Once you edit the properties file, use the NSCONFIG command to validate the NameServer configuration information in the file. If the file contains any other Unified Broker configurations, run the configuration validation utilities for those Unified Broker products to ensure that these configurations are still valid. For more information, see the "Using a Text Editor and Configuration Utilities" section.

The NSCONFIG utility displays the property settings associated with a NameServer configuration and checks that the syntax and values are valid. You must run the NSCONFIG utility locally on the machine on which the NameServer is running. The utility does not run across the network.

This is the syntax to invoke the NSCONFIG utility:

SYNTAX

```
nsconfig
  [
     [
        [ -name name-server ]
        [ -propfile path-to-properties-file ]
        [ -validate ]
     ]
   | -help
  ]
```

`-name` *name-server*

Specifies which existing NameServer configuration to examine. The name must match the name of an existing NameServer configuration in the specified properties file. If you do not specify a NameServer, the NSCONFIG utility analyzes all NameServer configurations defined in the properties file specified by the `-propfile` parameter.

`-propfile` *path-to-properties-file*

Specifies a filename or pathname to a file that contains the property settings to be validated, for example `test.properties`. If a filename or pathname is not specified, it defaults to the installation version of the `ubroker.properties` file, such as:

- `%DLC%\properties\ubroker.properties` on NT
- `$DLC/properties/ubroker.properties` on UNIX

`-validate`

Checks the syntax and values of property settings defined in the specified properties file.

`-help`

Displays command-line help.

For example, the following command validates the syntax and views the configurations of all NameServer instances defined within the `test.properties` file located in the current working directory:

```
nsconfig -propfile test.properties -validate
```

Once you have configured the controlling NameServer, you can configure your Unified Broker product components. You can configure each Unified Broker product much like the NameServer. For more information, see your product documentation.

8.3 Setting Up the Progress Unified Broker Environment

You might need to perform the following tasks before you start a Unified Broker instance:

- Set the standard Progress environment variables (for example, DLC) on the Unified Broker machine.
- Decide where to put your Unified Broker log files.
- Perform other preparations, depending on your product and configuration.

8.3.1 Setting Environment Variables

The environment variable settings required by a Unified Broker product vary according to the product. You can set them during product installation or when you configure the product. You can override most of these settings using the Progress Explorer or by editing the properties file (`ubroker.properties`).

Working Directory Settings

The properties file initially references a default environment setting for the working directory that you specify during installation. On UNIX, this is set in the `proadsv` script used to start the AdminServer and referenced in the properties file using `$WRKDIR`. You can remove or modify these references in the properties file to establish your own working directory settings for each Unified Broker and NameServer instance.

General Environment Settings

For more information on Progress environment settings, see Chapter 5, "Setting Up Your Environment," and also see the *Progress Client Deployment Guide* for information on setting up the environment for any Progress clients you are using with your Unified Broker product. For information on other environment variables unique to your Unified Broker environment, see your product documentation:

- *Building Distributed Applications Using the Progress AppServer*
- **WebSpeed** — *WebSpeed Installation and Configuration Guide*
- **DataServer** — *Progress DataServer for ORACLE Guide*

8.3.2 Configuring Progress Unified Broker Log Files

A Unified Broker installation generates a total of three log files, which (by default) are stored in the working directory with the following default names, where *type* is the Unified Broker type (for example, AS for AppServer) and *instance-name* is the name of the Unified Broker or NameServer instance:

- *instance-name*.broker.log - Unified Broker log file set using the Progress Explorer or the brokerLogFile property in the [Ubroker.*type*.*instance-name*] section (for example, [Ubroker.AS.asbroker1]) of the properties file. This log file records the connection and management activity of the Unified Broker instance.

- *instance-name*.server.log - Server process log file set using the srvrLogFile property in the [Ubroker.*type*.*instance-name*] section (for example, [Ubroker.AS.asbroker1]) of the properties file. This log file records the activity of server processes that directly handle client requests for the Unified Broker instance.

- *instance-name*.NS.log - NameServer log file set using the srvrLogFile property in the [NameServer.*instance-name*] section of the properties file. This log file records the activity of the controlling NameServer for a Unified Broker instance.

For disk management reasons, you might want to specify a nondefault location for the log files used by a Unified Broker. After you decide where you want these log files to reside, you can remotely specify the location for each one in the Progress Explorer or by directly editing the ubroker.properties file. For more information, see the "Configuring Progress Unified Broker Components" section.

You can also view these log files remotely using the Progress Explorer or locally on UNIX using the system text file commands and utilities.

The *instance-name*.broker.log and NameServer *instance-name*.log files include:

- Information on when Unified Broker and NameServer processes start up and shut down.

- System messages and information up to a level of detail that you specify using the Progress Explorer or the loggingLevel property in the ubroker.properties file.

The *instance-name*.server.log file includes all trace messages and Progress system messages that are not assigned an output destination by the OUTPUT TO KEEP-MESSAGES statement.

Because log files receive Unified Broker and NameServer startup and shutdown messages, Progress system messages, and trace messages, they can grow quickly. Also, they do not truncate automatically and can grow indefinitely. To manage disk space, periodically trim the file with a text editor after archiving any information that you want to save.

8.3.3 Other Preparations Prior To Startup

Depending on you product, you might need to perform other tasks before starting a Unified Broker instance. For example, you might have to customize the executables that execute the 4GL for an AppServer or WebSpeed application. You typically need to do this only if your application configuration requires nondefault Progress configurable elements, such as a DataServer. For more information, see your product documentation.

8.4 Starting Progress Unified Brokers

You can start a Unified Broker instance using the Progress Explorer from a remote Windows machine or the appropriate command-line management utilities.

For more information on starting a Unified Broker instance using the Progress Explorer, see the *Progress Installation and Configuration Guide Version 9 for Windows* and the Progress Explorer online help. The rest of this section describes how to start up a Unified Broker instance using the management utilities that run on UNIX.

8.4.1 Using the Management Utilities To Start a Unified Broker Instance

Follow these steps to start a Unified Broker instance using the management utilities:

1 ♦ Make sure that the AdminServer process is running on each of the following machines:

- Where the Unified Broker instance you want to start resides
- Where all required NameServers reside
- Where any Progress databases or other DataServers reside that your Unified Broker instance needs to access

2 ♦ If they are not already running, use the NSMAN utility to start each NameServer that you need (more than one if using connection-level fault tolerance) to support your Unified Broker instance. For more information, see the "Starting and Managing a NameServer Using the NSMAN Utility" section.

3 ♦ Start each Progress Database or DataServer that your application requires. For more information on starting databases and DataServers, see Chapter 10, "Starting and Running Progress," or your product documentation.

4 ♦ Using the command-line management utility for your Unified Broker product, start the Unified Broker instance. For more information on starting and managing a Unified Broker instance, see your product documentation.

8.4.2 Starting and Managing a NameServer Using the NSMAN Utility

The NSMAN utility runs on both Windows and UNIX platforms. It allows you to invoke the following management functions on a local or remote NameServer instance:

- Start a NameServer.
- Check the operational status of the NameServer.
- Stop the NameServer.

Unlike the Progress Explorer, the NSMAN utility has no mechanism for viewing log files or deleting configured NameServer instances. If you want to view the NameServer log file or delete the NameServer instance, you must do it manually using operating system commands. To delete the NameServer, you must remove the entry for this NameServer instance in the NameServer properties file. For more information on managing log files, see the "Configuring Progress Unified Broker Log Files" section. For more information on accessing the NameServer properties file, see the "Editing the Properties File" section.

NOTE: Before you can delete a NameServer instance, you must stop the NameServer.

This is the syntax to invoke the NSMAN utility:

SYNTAX

```
nsman
  {
      { -name name-server
        {
             -kill
           | -start
           | -stop
           | -query
        }
        [
             -host host-name -user user-name
           | -user user-name
        ]
        [ -port port-number ]
      }
    | -help
  }
```

`-name` *name-server*

This parameter is required. It specifies the name of the NameServer.

`-kill`

Stops and removes the NameServer from memory, no matter what it is doing.

`-start`

Starts the NameServer.

`-stop`

Tells the NameServer to stop itself.

`-query`

Queries the NameServer for its status.

-host *host-name*

Specifies the name of the machine where the AdminServer is running. If a host name is not specified, it defaults to the local host name.

-user *user-name*

Specifies a user name and prompts for a password. A user name and password are required only when you use the -host parameter and specify a remote host name. If you specify a remote host name with the -host parameter but do not specify a user name with the -user parameter, you receive a prompt for a user-name and password.

Windows NT supports three different formats for *user-name*:

– A user name as a simple text string, such as "mary", implies a local user whose user account is defined on the local NT server machine, which is the same machine that runs the AdminServer.

– A user name as an explicit local user name, in which the user account is defined on the same machine that runs the AdminServer except the user name explicitly references the local machine domain, for example ."\mary".

– A user name as a user account on a specific NT domain. The general format is *Domain**User*, in which the *User* is a valid user account defined within the domain and the *Domain* is any valid NT Server, including the one where the AdminServer is running.

-port *port-number*

Specifies the port number of the machine on which the AdminServer is running. If a port number is not specified, it defaults to 20931.

-help

Displays command-line help.

For example, the following command starts a NameServer instance named NSReservations on a host named SMPflights using UDP port 30000 as user admin:

```
nsman -name NSReservations -host SMPflights -port 30000 -user admin -start
```

Note that any nsman command that specifies a user name, typically prompts for the password, as well.

8.5 Understanding Load Balancing

Load balancing is a feature that allows client connection requests to be distributed among multiple Unified Broker instances that support the same Application Service. Load balancing is a NameServer option that comes installed with some products (for example, the WebSpeed Enterprise Transaction Server) or that you must install as an option with others (for example, the AppServer). If you have load balancing with your product and you are using a NameServer, the NameServer assigns client connections to the appropriate Unified Broker instances based on weight factors that you specify.

If the weight factor that you specify for each Unified Broker instance is appropriate in relation to the others, the effect is to assign more connections to broker instances with greater resources, and thus to balance connection load among all the instances. You can set the load-balancing weight factor for each Unified Broker instance in the Progress Explorer or by editing the `priorityWeight` property in the `ubroker.properties` file.

Percentage Weight Factors

Properly specified, weight factors give some sense of the amount of work that an individual Unified Broker instance can handle. For example, Table 8–2 shows the effect of weight factors specified for three Unified Broker instances registered to support the same Application Service.

Table 8–2: Weight Factors Based On Percentage

Unified Broker Name	Weight Factor	% Of Time Selected
AS1	20	20
AS2	20	20
AS3	60	60

The selection algorithm used by the NameServer guarantees that AS1 and AS2 are each selected 20% of the time and AS3 is selected 60% of the time. Thus, if the sum of weight factors for all Unified Broker instances that support the same application adds up to 100, each weight factor specifies the exact percentage of time that the NameServer selects the given Unified Broker instance over time.

Arbitrary Sum Weight Factors

You can specify arbitrary weight factors as any sum of values (not necessarily 100), but the weight of each is always proportional to the sum, as shown in Table 8–3.

Table 8–3: Weight Factors Based On Arbitrary Sums

Unified Broker Name	Weight Factor	% Of Time Selected
AS1	2	2/7
AS2	2	2/7
AS3	3	3/7

Fail-over Weight Factor

You can also specify a fail-over weight factor of zero (0) for a Unified Broker instance that you want to accept connection requests when the NameServer finds no other Unified Broker instance available in the pool.

8.6 Understanding Server-level and Connection-level Fault Tolerance

By default, a Unified Broker instance relies on a single controlling NameServer to resolve client connection requests, and a single Unified Broker instance to provide services to the client. You can configure the controlling NameServer so that multiple NameServer instances are available to resolve any client connection request, thus providing connection-level fault tolerance. If your product supports load balancing, you can also configure a single NameServer to resolve each connection request using multiple Unified Broker instances that support the same Application Service, thus providing server-level fault tolerance.

Figure 8–5 shows the relationship between these configuration options.

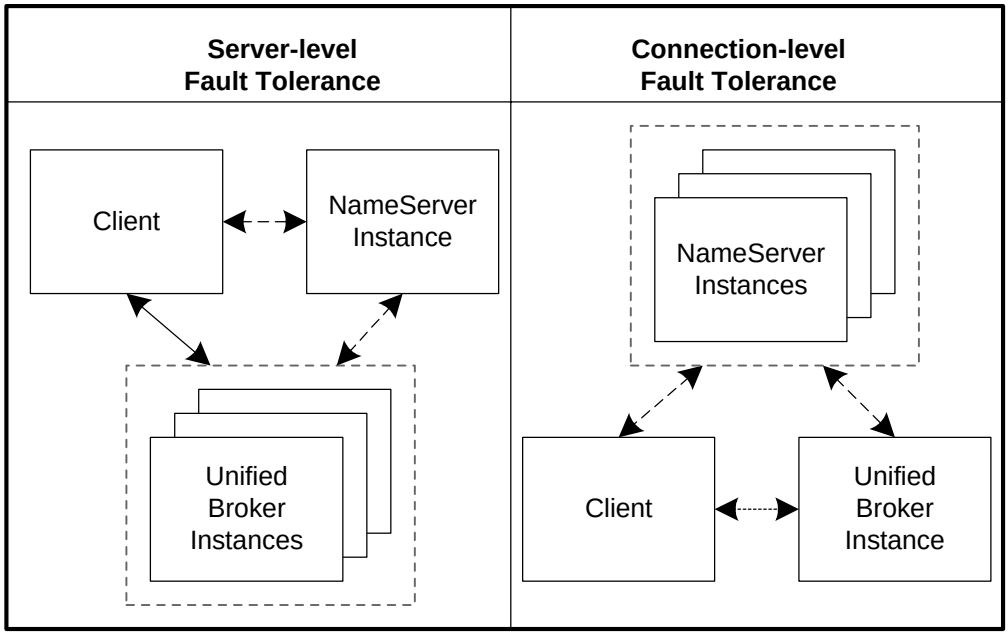


Figure 8–5: Server-level and Connection-level Fault Tolerance

These two levels of fault tolerance operate as follows:

1. **Server-level Fault Tolerance** — *Server-level fault tolerance* allows multiple Unified Broker instances to register with a NameServer for the same Application Service. A client requesting a connection is connected to one of several registered Unified Broker instances that the NameServer determines are available to provide the specified Application Service. If appropriate weight factors are specified, the NameServer also balances connection load among the several broker instances. For more information on load balancing, see the "Understanding Load Balancing" section.

2. **Connection-level Fault Tolerance** — *Connection-level fault tolerance* allows you to configure a collection of NameServers that work together to resolve a client connection request. You can use two different techniques, individually or together, to implement a fault-tolerant NameServer collection. This section describes these techniques and assumes that you are familiar with the documentation on using the NameServer with your Unified Broker product.

You can apply server-level and connection-level fault tolerance individually or together to achieve the level of fault tolerance that your application requires.

8.6.1 Connection-level Fault Tolerance

Connection-level fault tolerance enables a client (AppServer client, SonicMQ Adapter, WebSpeed Messenger, or DataServer client) to have its connection request satisfied by any NameServer from a set of related NameServers. You can configure NameServers for fault-tolerant operation using two different techniques, and you can use these techniques independently or together:

- **NameServer replication** — Where you configure multiple NameServer instances within a single subnet on different machines to listen on the same UDP port. Clients send connection requests and Unified Brokers send registration requests to all NameServer instances using UDP broadcasting. Using UDP broadcasting, the registration information from brokers is replicated on each NameServer that is listening (hence *NameServer replication*). Similarly, each client connection request is sent to each of the replicated NameServers.

- **NameServer neighbors** — Where you configure multiple NameServers on machines located in one or more subnets so that an initial NameServer instance receives the client connection request. If this initial NameServer cannot resolve the request, it passes the request on to a specified list of *NameServer neighbors*. These NameServer neighbors then attempt to resolve the connection request. Each NameServer neighbor represents the controlling NameServer for a separate Unified Broker instance.

Using either or both techniques, a client can get multiple responses from multiple NameServers. The client uses the first response that indicates that the requested Application Service was found. A client only gets a connection error if all NameServers that respond indicate that the Application Service cannot be found.

In general, you can combine NameServer replication with NameServer neighbors to provide connection-level fault tolerance across an entire network. The following sections describe how to implement connection-level fault tolerance using these techniques.

8.6.2 Using UDP Broadcasting

As described earlier, UDP is a connectionless protocol. This feature allows you to configure two types of communications with a NameServer:

- **Host request** — The client or Unified Broker sends a message directly to a NameServer residing on a specific host and listening on a specific port. The IP address represents the actual network location of a specific host. Only the NameServer on the specified host and listening on the specified port receives the message.

- **Broadcast request** — The client or Unified Broker sends a message specifying the UDP broadcast address of the NameServer host and the UDP port number on which the NameServer is listening. The UDP broadcast address represents the entire subnet where a host is located, and you can determine this address using the appropriate operating system commands from any host on the subnet. When a client or Unified Broker sends a UDP broadcast request, every NameServer on any host in the subnet that is listening on the specified port receives the message.

UDP broadcasting insulates the client and Unified Broker from having to know the exact host location of the NameServer. If there is some reason that you need to move the NameServer to a different machine in the same subnet, you can safely do it without having to change your client application or your Unified Broker configuration.

Figure 8–6 shows a client and Unified Broker using UDP broadcasting to communicate with the NameServer. Using the UDP broadcast address, 172.20.255.255, this client and Unified Broker can communicate with a NameServer running on any host in the 172.20 subnet.

Thus, you can use UDP broadcasting to support location transparency for a single NameServer. However, as Figure 8–6 implies, you can also use UDP broadcasting as the basis to support fault-tolerant NameServers using NameServer replication.

8.6.3 Using NameServer Replication

UDP broadcasting supports NameServer replication by allowing a client or Unified Broker request to be received by multiple NameServers listening on the same UDP port and configured on different machines within the same subnet. Because every host on a subnet receives every broadcast request, one or more of these hosts can support a NameServer that receives and handles the same messages. This provides fault tolerance for both a client connection request and a Unified Broker registration request.

Thus, to configure and use replicated NameServers, you must:

- Run each NameServer instance on a separate host located within the same subnet.
- Configure each NameServer instance to listen on the same UDP port.
- Configure each client application to send its connection request and each Unified Broker to send its registration request using the subnet UDP broadcast address instead of the NameServer host address.

There is one broadcast address for each subnet. Using this address and the specified UDP port number, a client or Unified Broker sends a single request that is recognized by every NameServer listening on that port in the subnet.

Figure 8–6 shows a client, a Unified Broker, and two replicated NameServers. The NameServer configurations shown for NameServer NS1 (above the dotted line) appear as they might in the `ubroker.properties` file for each host.

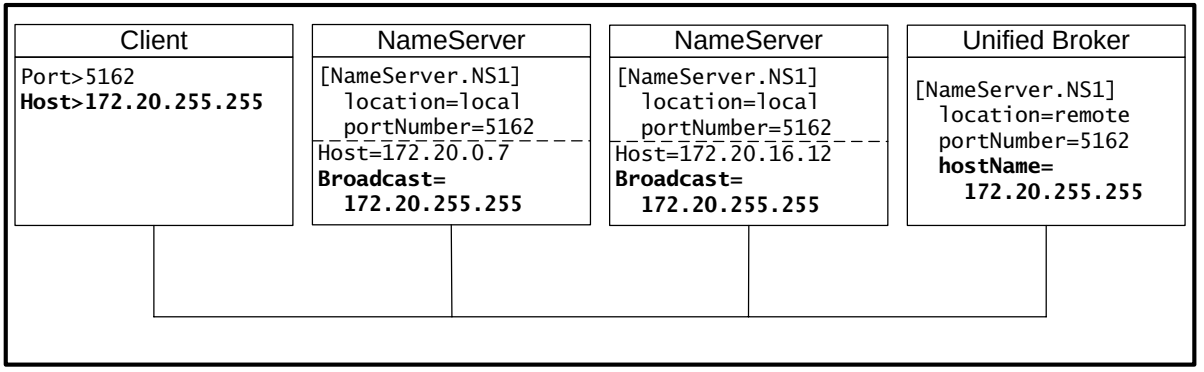


Figure 8–6: NameServer Replication

In Figure 8–6, one NameServer is located on a machine with the IP address 172.20.0.7 and another is located on a machine with the IP address 172.20.16.12. Both NameServers listen on UDP port 5162. The UDP broadcast address for these NameServers is 172.20.255.255. The Unified Broker is configured to register with a controlling NameServer remote from the Unified Broker machine using the UDP broadcast address 172.20.255.255 as the hostName. When the Unified Broker registers with its controlling NameServer using the UDP broadcast, it registers with both replicated NameServers. Similarly, when the client broadcasts its connection request using 172.20.255.255 as the NameServer host name, both replicated NameServers receive the request. The client uses the Unified Broker connection returned by the first NameServer that responds.

Note that if the NameServer at IP address 172.20.0.7 moves to a different host on the subnet, for example, with IP address 172.20.16.5, neither the client application nor the Unified Broker configuration has to change.

Implementing NameServer Replication

To configure and use NameServer replication, follow these general steps:

1 ♦ Install the NameServer on each host within a single subnet where you want to replicate a NameServer configuration.

2 ♦ Configure each replicated NameServer to listen on the same UDP port number.

3 ♦ Determine the UDP broadcast address for the subnet where the NameServer hosts reside. For more information, see the "Determining the Broadcast Address" section.

4 ♦ Configure each Unified Broker instance (AppServer, WebSpeed Transaction Server, or DataServer) to use a controlling NameServer as follows:

- **Location** — Remote
- **Host name** — The UDP broadcast address that you determined in Step 3
- **Port number** — The UDP port number that you specified in Step 2

5 ♦ Provide connection parameters to the client (AppServer, DataServer, or WebSpeed) that specify the required Application Service name, the broadcast address in Step 3, and the UDP port number that you specified in Step 2.

Determining the Broadcast Address

You can determine the broadcast address of a UNIX machine by using the `netstat` and `ifconfig` commands, as in this example:

```
$ netstat -i
Name  Mtu  Net/Dest   Address    Ipkts   Ierrs Opkts  Oerrs Collis Queue
lo0   8232 loopback    localhost   771334  0     771334 0    0     0
le0   1500 bali        bali       15069970 286170 10019158 1     302211 0
$ ifconfig le0
le0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
       inet 172.20.0.7 netmask ffff0000 broadcast 172.20.255.255
```

This shows that the IP address for bali is 172.20.0.7, and its broadcast address is 172.20.255.255.

On NT, follow these steps to determine the broadcast address:

1 ♦ Enter the `ipconfig` command in the console, as shown in this example:

```
C:\>ipconfig

Windows NT IP Configuration

Ethernet adapter CE2XPS1:

IP Address. . . . . . . : 172.18.103.44
Subnet Mask . . . . . . : 255.255.0.0
Default Gateway . . . . . : 172.18.0.19
```

2 ♦ For each bit in the Subnet Mask that has a value of 0, convert the corresponding bit in the IP Address to 1.

Note that the IP Address and Subnet Mask are composed of four dot-separated decimal numbers, and each decimal number represents an 8-bit binary number. Also note that the decimal number 255 is 11111111 in binary.

In this example, the last two decimal digits of the Subnet Mask are zeros. Since the corresponding bits in the IP Address must be converted to 1, the last two decimal numbers of the IP Address should be 255. Therefore the broadcast address is 172.18.255.255. (For more information on determining broadcast addresses, consult with you network administrator.)

8.6.4 Using NameServer Neighbors

In a typical environment where UDP broadcasting is used, there is at least one NameServer in each subnet where a Unified Broker exists. A client application that wants to make use of Unified Brokers in each subnet can make a separate connection request to the appropriate controlling NameServer for each Unified Broker. However, NameServer neighbors allow the client to make all of its connection requests using a single NameServer address.

NameServer neighbors are alternate NameServers that you specify as part of a NameServer configuration. When a NameServer receives a connection request from a client that it cannot resolve, it automatically passes the request to the specified NameServer neighbors to attempt the resolution.

NameServer neighbors support client applications in a similar way to replicated NameServers in that the client uses the first response returned by a NameServer indicating that the requested Application Service was found. However, unlike replicated NameServers, each NameServer neighbor is typically the controlling NameServer for a separate and distinct Unified Broker configuration that might not support the same Application Services as the others.

You can configure each NameServer neighbor as a NameServer with its own set of NameServer neighbors. Thus, you can link NameServer neighbors to other NameServer neighbors for an arbitrary level of depth. You can also replicate both initial NameServers and NameServer neighbors for maximum fault tolerance.

Follow these steps to configure NameServer neighbors:

1 ♦ Define the NameServer neighbors as local NameServer instances on your network by using the Progress Explorer or by editing the `ubroker.properties` file. These neighbor NameServers can be defined in the same or different subnets in your network.

2 ♦ Specify their names as NameServer neighbors when you define the NameServer in the Progress Explorer or assign the names in a comma-separated list as the value of the `neighborNameServers` property for the definition in the `ubroker.properties` file.

Figure 8–7 shows how you can do this. This example combines NameServer replication with NameServer neighbors.

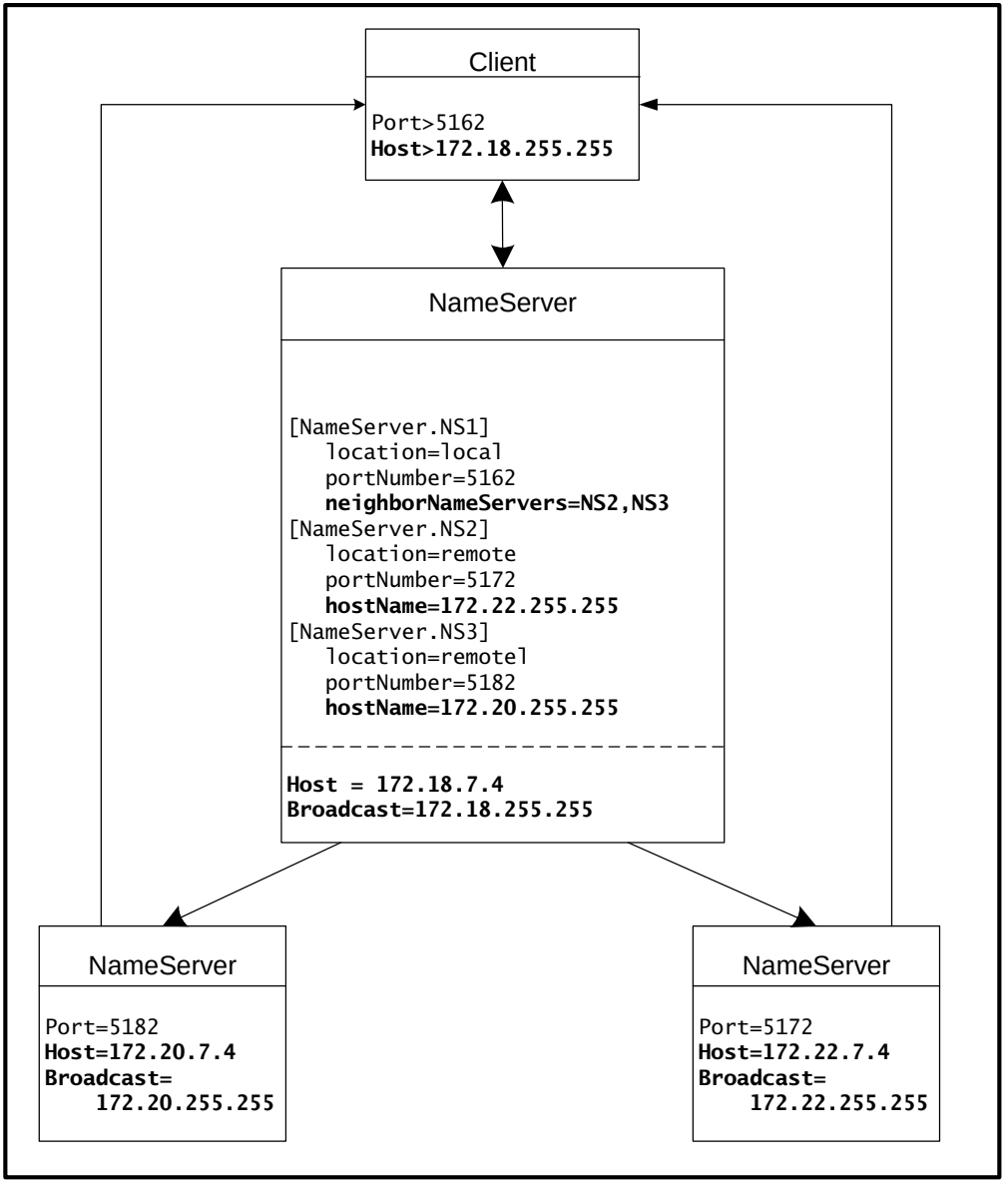


Figure 8–7: NameServer Neighbors

In Figure 8–7, if a client requests a connection to a Unified Broker that supports the Inventory Application Service using the 172.18.255.255 broadcast address, the NameServer at host 172.18.7.4 receives the request. If this NameServer has a Unified Broker that registered for the Inventory Application Service, it returns the location of that Unified Broker back to the client. If it does not have a Unified Broker that registered for the Inventory Application Service, this NameServer forwards the request to its neighbors, specified using the broadcast addresses 172.20.255.255 and 172.22.255.255. In this instance, the NameServers at hosts 172.20.7.4 and 172.22.7.4 receive the request.

NOTE: If you replicate these NameServers, all of the replicated NameServers in each subnet receive the request.

Neither of these NameServers have neighbors of their own, so both of them send a response back to the client. It does not matter if one of the NameServers does not know about the requested Application Service. The client uses the first positive acknowledgement from a NameServer and disregards the rest. The client application only receives an indication that the Application Service was not found if all responding NameServers indicate that the Application Service was not found.

Note that while NameServer neighbors provide the most benefit when using UDP broadcast, there is no requirement to do so. The hostName properties for NS2 and NS3 in Figure 8–7 can explicitly specify 172.20.7.4 and 172.22.7.4. You might want to use NameServer neighbors without broadcasting when you must tie together preconfigured NameServers, but where the performance implications of broadcasting outweigh the benefits. For more information, see the "Performance Implications Of Broadcasting" section.

8.6.5 Performance Implications Of Broadcasting

When you use UDP to send a specific host request, only the specified host examines the message to determine what port it was sent to and whether an application (like the NameServer) is running and listening to that port on the host. However, when you use UDP broadcasting, either for NameServer replication or only to provide location transparency for a single NameServer, every host in the specified subnet examines the message for this same information.

Thus, using UDP broadcasting might have a significant impact on the performance of your network if you have a large number of client applications that frequently connect to Unified Brokers. In deciding whether to use UDP broadcasting, you must weigh the benefits of location transparency for a single NameServer or replication of multiple NameServers against the impact on your Unified Broker and network performance.

9

Maintaining Two Versions

This chapter provides information about maintaining Version 9 and either Version 7 or Version 8 of Progress on your system.

Specifically, this chapter contains the following sections:

- Maintaining Two Versions
- Running Multiple AdminServers

9.1 Maintaining Two Versions

Follow these steps to maintain Version 9 and either Version 7 or Version 8 of Progress on your UNIX or Linux system:

1 ♦ Follow the procedure for installing a new product. When the installation procedure prompts you for a Select Destination pathname, type a directory other than the one in which the current product is installed. For example, you can use the Progress version number `/usr/dlc83a` or `/usr/dlc91c`.

2 ♦ When prompted, instruct the installation procedure not to copy the scripts to `/usr/bin`.

3 ♦ In Version 9.1 you can use proenv to set DLC bin in your path. Enter the following command at the system prompt to run proenv:

```
installation-path/bin/proenv
```

This command will spawn a new shell and generate the proenv prompt in your current window. Proenv automatically sets the DLC environment variable in your path.

4 ♦ To access the new product, set the environment variables PATH, PROPATH, and PROMSGS.

If you are running the Bourne shell or the Korn shell, put the following lines in your `.profile` file or type them either at the system prompt or in the proenv window:

```
PATH=$DLC/bin:$DLC:$PATH; export PATH
PROPATH=$DLC:$PROPATH; export PROPATH
PROMSGS=$DLC/promsgs; export PROMSGS
```

If you are running the C shell, put the following lines in your `.login` file or type them either at the system prompt or in the proenv window:

```
set PATH=(${DLC}/bin ${DLC} ${PATH})
setenv PROPATH "$DLC $DLC/bin"
setenv PROMSGS=$DLC/promsgs
```

The *new-directory-name* is the name of the directory where you installed the new product.

Include the DLC variable at the beginning of the definition of the PATH variable to ensure that UNIX first looks in the directory defined by the DLC variable before searching `/usr/bin` for the correct startup scripts. Set the DLC variable to ensure that the scripts look for the correct version of the product when they execute.

5 ♦ To access Progress Version 7 or Version 8, set the DLC and PATH environment variables of the earlier version to the directory of the new version while running the new version.

6 ♦ If you entered changes in your `.profile` or `.login` file, you must log out and back into the system for your changes to take effect.

9.2 Running Multiple AdminServers

If you are running multiple AdminServers you need to override both the default port and the default adminport settings. See "Using the AdminServer" in Chapter 5, "Setting Up Your Environment," for more information on changing the AdminServer port and adminport settings.

10

Starting and Running Progress

This chapter describes how to start up and run a Progress database.

Specifically, it contains the following sections:

- Starting Progress
- Running Progress Clients and Servers On a Network

10.1 Starting Progress

Progress startup commands differ with certain operating systems, user interfaces, and network software. UNIX provides a series of scripts to run the Progress executables, such as `proserve` to start broker/servers and `mpro` to start multi-user interactive clients. These scripts are tailored for your particular software environment. For information on the script executed by each command, see the description of the command in Table 10–1.

It is important that you observe the following conventions when you enter a command:

- Use lowercase characters for commands on UNIX.
- Enter parameters on UNIX exactly as shown in the syntax descriptions.
- Values can be case sensitive on UNIX. (For example, names of UNIX files are case sensitive.)

10.1.1 Startup and Shutdown Commands

Startup commands start a Progress session and connect you to a database. The following tables summarize the startup commands for each operating system and their functions. For detailed information on these commands and their parameters, see the descriptions of the commands later in this chapter.

Table 10–1 describes each of the command components.

Table 10–1: Progress Command Components

Component	Description
command	On UNIX, the command runs a script that executes a Progress executable with appropriate parameters.
db-name	Name of the database you want to connect to.
parameter, qualifier	Operating criteria for the command.
value	Numeric value or file specification for the parameter.

Table 10–2 summarizes the tasks you can perform and the related startup and shutdown commands to use on UNIX systems.

Table 10–2: UNIX Startup and Shutdown Commands *(1 of 2)*

Task	Command
Start a UNIX character Progress Procedure Editor and connect to a single-user database.	`pro db-name`
Start a UNIX character Progress Procedure Editor and connect to a multi-user database.	`mpro db-name`
Start a UNIX Progress character client session in batch mode and connect to a single-user database.	`bpro db-name` `-p procedure-name`
Start a UNIX Progress character client session in batch mode and connect to a multi-user database.	`mbpro db-name` `-p procedure-name`
Start a Progress server-group.	`proserve` `-servergroup server-group-name`

Start a server or broker for a multi-user Progress database.	`proserve db-name`
	`-S service-name`
	`-H host-name`
	`-N network-type`

Shut down a multi-user server or broker for a Progress database.	`proshut db-name`

Start a remote Progress DataServer broker.	`probrkr -S service-name`
	`-H host-name`
	`-N network-type`

Start an asynchronous page writer (APW) for a database[1].	`proapw db-name`
Start a before-image writer (BIW)[1].	`probiw db-name`
Start an after-image writer (AIW)[1].	`proaiw db-name`
Start the Progress Watchdog utility[1].	`prowdog db-name`

Table 10–2: UNIX Startup and Shutdown Commands *(2 of 2)*

Task	Command
Shut down a remote Progress DataServer.	proshut *db-name* -S *service-name* -H *host-name* -N *network-type*
Shut down an APW, AIW, BIW, or Watchdog process.[1]	proshut *db-name* Choose option 1 (Disconnect a User) to disconnect the process.

[1] Option available only on Enterprise product.

10.1.2 Starting Single-user Progress In Interactive Mode

To start single-user Progress, enter the following command:

```
pro [ db-name ] [ parameters ]
```

db-name

Specifies the database you want to start (-db is implicit but can be specified).

parameters

Specifies the startup parameters you want to use to describe system and application characteristics. For a detailed description of the Progress startup parameters, see the *Progress Startup Command and Parameter Reference*.

10.1.3 Starting Single-user Progress In Batch Or Background Mode

Batch or background processing is convenient for large-scale database updates or procedures that you can run unattended (at night, for example).

To start single-user Progress in batch or background mode, enter the following command:

```
bpro [ db-name ] -p procedure-name
  [ parameters ] > output-file
```

db-name

Specifies the database you want to start.

-p *procedure-name*

Specifies the procedure to run at startup.

parameters

Specifies the startup parameters you want to use.

output-file

Specifies the name of the file that receives all output to the default stream.

Redirecting Output

On UNIX you can redirect batch job input and output with the greater than (>) and less than (<) redirection symbols. You can also use the pipe symbol (|) to put a Progress batch run in a command pipeline. See the Batch (-b) startup parameter in the *Progress Startup Command and Parameter Reference* for more information.

In the following example, Progress starts in batch or background mode against the sports database and automatically runs the sportsbat startup procedure. In addition, the system directs output (not otherwise directed) with an OUTPUT TO statement to the file named `errlist`:

```
bpro sports -p sportsbat.p > errlist
```

10.1.4 Starting the Multi-user Server Or Broker

Before you can run multi-user Progress, you must start the multi-user server process. The server process coordinates all the database requests from all the users using a single database. Enter the following command to start the multi-user server process:

NOTE: On UNIX systems that run shared-memory versions of Progress, the main database server is called the *broker*. The broker process manages shared resources and starts servers for remote users, if necessary. For more information, see the "Shared-memory Architecture" section in Chapter 7, "Configuring Progress."

```
proserve db-name [ parameters ]
```

db-name

Specifies the database you want to start Progress against (-db is implicit).

parameters

Specifies the startup parameters for the broker/server. See the *Progress/400 Product Guide* for a list of broker/server startup parameters.

10.2 Running Progress Clients and Servers On a Network

After setting up your database on the network, you are ready to run Progress. In general, the procedure for running clients and servers on a network of systems is very similar to the procedure for running them on a single system. First, you must start all required database servers and application servers; then, start the client sessions that connect to them.

10.2.1 Using Network Startup Parameters

To connect network clients, servers, and application servers, you might have to use a variety of startup parameters to establish and manage network communications among them. The requirements and use of these parameters vary on different operating systems and network environments. For more information on using startup parameters, see the *Progress Startup Command and Parameter Reference*.

Client Network Parameters

Table 10–3 lists the parameters you use to supply Progress clients with necessary network information.

Table 10–3: Client Parameters

Parameter	Syntax
Host Name[1]	TCP: -H *host-name* SNA: -H PLU=plu-*name*
Message Buffer Size	-Mm *n*
Network Type	-N *network-type*
Service Name	-S *servicname port-number*

[1] For the TCP network type, this required parameter specifies the machine name (address) where the server runs.

Server Network Parameters

Table 10–4 lists the parameters you use to supply Progress brokers and servers with necessary network information. In a Progress AppServer configuration, use the same parameters to pass information to application brokers and application servers. For more information on using server parameters, see the *Progress Startup Command and Parameter Reference*.

Table 10–4: Server Parameters *(1 of 2)*

Parameter	Syntax
Host Name	TCP: -H *host-name* SNA: -H PLU=plu-*name*
Manual Server	-m2
Secondary Login Broker	-m3
Maximum Clients Per Server	-Ma *n*
Minimum Clients Per Server	-Mi *n*
Maximum Dynamic Server	-maxport *n*

Table 10–4: Server Parameters *(2 of 2)*

Parameter	Syntax
Minimum Dynamic Server	-minport *n*
Maximum Servers	-Mn *n*
Maximum Servers Per Protocol	-Mp *n*
Maximum Servers Per Broker	-Mpb *n*
Network Type	-N *network-type*
Service Name	-S *servicename port-number*

For more information on the syntax and values for each parameter, see the *Progress Startup Command and Parameter Reference*.

10.2.2 Specifying the Network Type (-N)

Each Progress executable has a default network type determined by the operating system on which it runs. Table 10–5 lists the default network type that Progress uses on each supported operating system.

Table 10–5: Default Network Types

Operating System (Executable)	Default Network Type
Windows (client or server)	TCP
UNIX (client or server)	TCP
AS/400 (server)	SNA

10.2.3 Network Addressing With Progress (-S and -H)

In all network environments, you use the Service Name (-S) startup parameter to assign a name to a Progress broker/server. You then address this broker/server from a remote client by using the same value for -S as a startup or database connection parameter. Depending on your network type, you might also have to specify additional addressing criteria for remote Progress clients. In terms of Progress addressing, the TCP protocol uses host addressing.

The TCP protocol requires a remote client to explicitly address the database server machine (or host) on which the server runs. In a TCP network, you must use the Host Name (-H) startup parameter to specify the host address. The -H value is the name assigned to the database server machine in your TCP/IP hosts file.

10.2.4 Starting Applications On a Network

In general, you can start a Progress application on a network by following these steps:

1 ♦ Start each broker or server on its database server machine or AppServer machine.

2 ♦ Start the client applications on the application workstations.

Starting Network Brokers and Servers

You can start most network brokers and servers using the PROSERVE command for your database server machine. For example, on a UNIX machine you enter the following command to start brokers for two databases (sports and news) using the TCP network type:

```
proserve sports -S spsrv -N TCP -db news -S nwsrv -N TCP
```

Starting TCP/IP Clients On UNIX

You can start most network clients using the MPRO command for your application workstation. For example, on UNIX machines you can enter the following command to start a client application named `spapp.p` for two databases (sports and news) managed on a host named `dbmach` using the TCP network type:

```
mpro sports -p spapp.p -S sportssv -H dbmach -N TCP
     -db news -S newssv -H dbmach -N TCP
```

10.2.5 Starting Multiple Brokers Using the Same Protocol

You can start multiple brokers that use the same protocol. The -Mn parameter and a new parameter, Maximum Servers per Broker (-Mpb), determine the number of servers a broker can start.

Use the following commands to start two brokers that use TCP and start multiple servers each:

```
proserve db-name -S server-name -N network-type
  -H host-name -Mn n -Mpb n
proserve db-name -S server-name -N network-type
  -H host-name -Mpb n -m3
```

db-name

Specifies the database you want to start. If the database is not in the current directory, you must specify the full pathname of the database.

-S *service-name*

Specifies the database server or broker process service name. You must specify the service name in a TCP network.

-N *network-type*

Specifies the network protocol, which is TCP.

-H *host-name*

Specifies the machine where the database server runs.

-Mn *n*

Specifies the maximum number of remote client servers and login brokers that the broker process can start.

-Mpb *n*

Specifies the number of servers that the login broker can start to serve remote users. This applies to the login broker that is being started.

-m3

Starts the secondary login broker.

For example, you would use the following commands to start two brokers that use TCP and start four servers each:

```
proserve db -S demosv1 -N tcp -H myhost -Mn 9 -Mpb 4
proserve db -S demosv2 -N tcp -H myhost -Mpb 4 -m3
```

As the example shows, the -Mn value must be large enough to account for each additional broker and all servers. If you do not specify -Mpb, the value of -Mn becomes the default.

You must include the -m3 parameter with every secondary broker startup command. While the (-Mpb) parameter sets the number of servers a broker can start, the -m3 parameter actually starts the secondary broker.

If you start multiple brokers, you should also run the Progress Watchdog process (PROWDOG). PROWDOG enables you to restart a dead secondary broker without shutting down the database server.

10.2.6 Accessing a Server Behind a Firewall

Progress allows you to use the Minimum Dynamic Server Port (-minport) and the Maximum Dynamic Server Port (-maxport) server startup parameters to provide client access to a Progress server that is behind a firewall. This communication is possible only when the access to the server can be limited. You supply this limit when you specify a group of port numbers with the -minport and -maxport parameters.

For example, suppose you start the following two login brokers:

```
proserve db -S demosv1 -N tcp -H myhost -minport 4000 -maxport 4040
proserve db -S demosv2 -N tcp -H myhost -minport 4041 -maxport 4080 -m3
```

A client requesting a connection from the first broker, demosv1, is assigned a port number in the range of 4000 to 4040. The 4000-to-4040 range limits access to the server by limiting communication to just 40 ports.

The default for -minport is 1025 for all platforms. Ports lower than 1025 are usually reserved for system TCP and UDP. The default for -maxport is 2000 for all platforms. Remember that some operating systems choose transient client ports in the 32768-to-65535 range. Choosing a port in this range might produce unwanted results.

10.2.7 Starting and Running Multi-user Progress In Interactive Mode

Enter the following command to start and run Progress in interactive mode:

```
mpro [ db-name ] [ parameters ]
```

db-name

Specifies the database you want to start. If the database is not in the current directory, you must specify the full pathname of the database.

parameters

Specifies the startup parameters you want to use.

On UNIX, the MPRO command starts either a local or remote client. If the Host Name (-H) and Service Name (-S) parameters are supplied, Progress starts a remote client—a client that is assigned to a server. Otherwise, Progress starts a local self-service client. Note that specifying -H and -S when starting a client on the local host machine actually produces a “local remote client” (a local process that accesses the database through a server.)

The database you name when starting multi-user Progress must be in the current directory, or you must specify the full pathname of the database. For example, if you are using UNIX and you log in as sue, the login directory is `/usr/sue`.

10.2.8 Starting and Running Multi-user Progress Clients In Batch Or Background Mode

Before you can start a multi-user Progress batch or background job, you must start the server for the database you want to use.

To start multi-user Progress in batch or background mode, enter the following command:

```
mbpro db-name -p procedure-name [ parameters ]
 > error-file
```

db-name

Specifies the database you want to start.

-p *procedure-name*

Specifies the procedure to run at startup.

parameters

Specifies the startup parameters you want to use.

error-file

Specifies the file where error messages are sent.

A

Terminals

This appendix provides a list of built-in terminal definitions supplied with Progress.

Specifically, it contains the following sections:

- Terminal Issues
- Terminal Identifiers

A.1 Terminal Issues

When you start Progress, you might receive the following message:

```
** You cannot use DEL for both stty intr and DELETE-CHARACTER.
```

This message indicates that you were trying to use the **DEL** key as the UNIX interrupt key and as the Progress DELETE-CHARACTER key. To avoid this message, add the following line to your `.profile` file:

```
stty intr ^C
```

This command resets your UNIX interrupt key from **DEL** to **CTRL-C**.

Built-in terminal definitions are supplied with Progress for the terminals listed in Table A–1, which indicates the terminal identifiers you can use so that Progress can successfully access that terminal definition. Be sure the operating system environment variable TERM is set to the appropriate value. For example:

```
TERM=wyse60;export TERM
```

A.2 Terminal Identifiers

This section lists terminal identifiers, as seen in Table A–1.

Table A–1: Terminal Identifiers *(1 of 2)*

Terminal Model	Terminal Identifier	Notes
xterm	xterm	–
CDE dtterm	dtterm	–
DEC VT100	V1, vt100, vt100-80	Asian languages are supported. For more information on supported languages, see the *Progress Internationalization Guide*.

Table A–1: Terminal Identifiers *(2 of 2)*

Terminal Model	Terminal Identifier	Notes
DEC VT200 Series	V2, vt200, vt200-80, vt220, vt220-80, vt240, vt241	Asian languages are supported. For more information on supported languages, see the *Progress Internationalization Guide*.
DEC VT300 Series	V3, vt300, vt320, vt330, vt340, pt300, pt-100	–
DEC VT400 series	V4, vt400, vt400-80, vt420	–
DEC VT500 series	V5, vt500, vt500-80, vt510, vt520, vt525	–
IBM 3151	3151, m3, ibm3151	–
IBM PC/AT XENIX Console	li, ansi	Ansi driver.
IBM PC/AT XENIX Color Console	lc, ansic	Ansi driver. Uses reverse video for input fields.
Linux Console	linux, linux-lat	–
Sun Console	Mu, sun	–
Wyse 60	w6, wy60, wyse60 wy60t, wyse60tall wy60w, wyse60wide wy60tw, wyse60tall + wide	Assumes that the function keys are set to the factory defaults. Check the PROTERMCAP entry for setup mode. Terminal in following mode: 43 lines X 80 columns. 25 lines X 132 columns. 43 lines X 132 columns.
Wyse 370	wy370, wyse370	–

NOTES:

- The IBM native console terminal type hft is not supported.
- Progress Version 9 does not support spacetaking terminals, unless the terminal has a firmware setup option to change it to a nonspacetaking mode.

Table A–1 is complete as of the print date of this guide. If your terminal is not among those listed, then follow these steps:

1 ♦ Try to run Progress using a terminal definition for a terminal that functions similarly to yours, or try to configure your terminal to emulate one of the supported terminals.

NOTE: Progress Software Corporation does not support terminal emulation.

2 ♦ In the directory where you installed your Progress product, find the PROTERMCAP file that contains terminal definitions. Search through the PROTERMCAP file to see if your terminal is listed. The PROTERMCAP file is similar in structure to the UNIX `/etc/termcap` file. Each terminal type is followed by a description of that terminal. For more information about the PROTERMCAP file, see the *Progress Client Deployment Guide*.

B

Progress National Language Support

This appendix gives information about Progress messages in different languages. Specifically, this appendix contains the following sections:

- Packaging
- Directory Structure
- Contents Of Each Directory
- Implementing Regional Support
- International Databases
- Progress Messages
- Environment Variables Of the SQL-92 Client
- Regional Parameter Files

B.1 Packaging

Table B–1 provides a list of the PROMSGS translations shipped to all Progress users by default.

Table B–1: PROMSGS Translations Shipped With Progress

Supported Languages	
Arabic (ARB)	Korean (KOR)
Chinese-Simplified (SCH)	Lithuanian (LIT)
Chinese-Traditional (TCH)	Norwegian (NOR)
Croatian (HRV)	Polish (POL)
Czech CZE)	Portuguese (POR)
Danish (DAN)	Portuguese-Brazilian (BRZ)
Dutch (DUT)	Romanian (ROM)
English-American (AME)	Russian (RUS)
English-International (ENG)	Serbian (SRB)
Finnish (FIN)	Slovak (SVK)
French (FRE)	Slovenian (SVN)
German (GER)	Spanish (SPA)
Greek (GRE)	Spanish-Latin American (SPL)
Hebrew (HBR)	Spanish-Mexican (MEX)
Hungarian (HUN)	Swedish (SWE)
Icelandic (ISL)	Thai (TAI)
Italian (ITA)	Turkish (TUR)
Japanese (JPN)	–

B.2 Directory Structure

The `$DLC/prolang/Readme` file lists the subdirectories of the `/prolang` directory by language. Also included is helpful information about code-page tables in `convmap.dat`.

B.3 Contents Of Each Directory

The `prolang` directory contains a subdirectory for each national language that Progress supports. In each of these directories, there can be five file types. The variable *`lang`* stands for language-specific references. Table B–2 provides a description of each file type.

Table B–2: National Language File Descriptions

File Name	Description
`empty.db`	An empty, language-specific Progress directory containing databases of various block sizes. The database is initialized with an appropriate code page and collation for your language.
`promsgs.`*`lang`*	A translated Progress run-time messages file.
`lang``.pf`	A file containing the parameters used to start up Progress with the appropriate settings for your region.
`lang``.df`	A data definition file that can be loaded into an empty Progress database to create a language-specific database. A database created by loading this file is identical to the empty database provided in this directory. You can use this file to create sort ordering variations in the database. In the Asian directories, the file is named `_tran.df`.

B.4 Implementing Regional Support

The installation utility requires you to install at least one language.

During the installation you must choose a default language. If you want to change the default language after installing Progress, see the *Progress Internationalization Guide* for detailed instructions.

The International Settings dialog box of the installation program creates a Progress Startup (`startup.pf`) file to accommodate international conventions such as Date format, Number format, Character set, Collation, and Case.

Once you select the default language, the installation program copies the contents of the DLC/prolang directory to $DLC. This affects your empty.db and promsgs files.

See the *Progress Internationalization Guide* for more information about any of the following files:

- empty.db
- startup.pf
- promsgs

B.5 International Databases

As part of the installation media, Progress supplies empty databases that support the language and collation standards of over thirty languages. The databases are located in the $DLC/prolang subdirectories. See Table B–1 for the subdirectory name for your language. For example, the empty database that you might use to build a Russian application is $DLC/prolang/rus/empty.db.

These empty databases provide a database labelled with the appropriate code page and collation table for a language. However, if you are developing applications for a language or region that is not represented in $DLC/prolang, Progress has a utility, PROUTIL, that you can use to set up a unique database. See the *Progress Internationalization Guide* for detailed instructions.

B.6 Progress Messages

The text used in Progress messages is contained in the promsgs file. Progress provides various language editions of the promsgs file in the $DLC/prolang subdirectories that you select during installation. Each file has an extension that identifies its language.

To run Progress with a certain version of promsgs, set the PROMSGS environment variable to the appropriate file. By default, Progress displays messages from $DLC/promsgs.

B.7 Environment Variables Of the SQL-92 Client

Progress Version 9.1 contains the environment variables SQL_CLIENT_CHARSET and SQL_CLIENT_CHARSET_PROMSGS for SQL-92 clients. You can use these variables to internationalize your applications. These environment variables determine the code page the client uses to display the following:

- Database data from the server
- PROMSGS from the server

NOTE: Set SQL_CLIENT_CHARSET only if you want clients to use a code page that is different from the code page the client operating system uses. Set SQL_CLIENT_CHARSET_PROMSGS only if you want run-time messages to use a code page that is different from either the code page the client operating system uses, or the code page set by SQL_CLIENT_CHARSET.

If you do not set either of these environment variables, then the SQL-92 client code page will correspond to the language of the client operating system.

Code Page Client Uses To Display Data

To display database data from the server, the client uses the code page set by SQL_CLIENT_CHARSET, if you have set this environment variable. Otherwise, the client uses the code page of the client's operating system.

If you want to specify an SQL-92 client code page that is different from the client operating system, you can set the SQL_CLIENT_CHARSET environment variable to the name of a Progress code page. When you set this variable to a code page, the SQL-92 server converts text data that is sent from the server to the client to the code page set by SQL_CLIENT_CHARSET. The server also converts text data that is sent from the client to the server to the server code page.

Code Page Client Uses To Display PROMSGS From the Server

To display PROMSGS from the server, the client uses the code page set by SQL_CLIENT_CHARSET_PROMSGS, if you have set this environment variable. Otherwise, the client uses the code page set by SQL_CLIENT_CHARSET, if you have set this environment variable. If you have set neither of these environment variables, then the client uses the code page of the client's operating system.

If you want run-time messages at the SQL-92 client to use a different code page from either the client operating system or the code page set by SQL_CLIENT_CHARSET, you can set the SQL_CLIENT_CHARSET_PROMSGS environment variable. When you set this variable to a code page, the SQL-92 server converts run-time messages that are sent from the server to the client to the code page set by SQL_CLIENT_CHARSET_PROMSGS.

NOTE: The SQL_CLIENT_CHARSET_PROMSGS environment variable applies to SQLDUMP and SQLLOAD, which are actually SQL-92 applications — specifically, ESQL/C applications.

B.8 Regional Parameter Files

A useful technique for controlling a Progress client session or server is to use a parameter file (.pf) with a startup or connection command. Progress provides parameter files that set up Progress sessions appropriately for a wide range of countries. You can use .pf files to specify the correct code-page settings for international applications. The setup of the *install-path*`/startup.pf` file is based on the installation options that you select.

The international parameter files are located in the `$DLC/prolang` subdirectories. Parameter files are region- or country-specific rather than language-specific because parameter files set options that can vary from country to country. The `$DLC/prolang/ger` directory has parameter files for Austria, Germany, and Switzerland to account for the differences among these German-speaking countries.

You should use the parameter file to make sure that the application and database are using the appropriate international settings. Typically, a parameter file for a internationalized application sets the parameters listed in Table B–3.

Table B–3: Startup Parameters For a Deployed Application *(1 of 2)*

Parameter	Description
Internal Code Page (-cpinternal)	The code page that Progress uses in memory.
Stream Code Page (-cpstream)	The code page for stream I/O.
Case Code Page (-cpcase)	A case table in the `convmap.cp` file to use for uppercase/lowercase rules. Case rules are used by the CAPS and LC functions and by the ! formatting character.
Collation Code Page (-cpcoll)	A table in the `convmap.cp` file to use for collation rules.

Table B–3: Startup Parameters For a Deployed Application *(2 of 2)*

Parameter	Description
Date Format (-d)	The format in which an application displays dates. Specify the format as a three-character string, comprised of the letters d, m, y, in the order that you display the date.
Language (-lng)	The initial value for the CURRENT-LANGUAGE function, which determines from which r-code segment Progress reads character-string constants. Specify the language as a character string in quotes.
European Numeric Format (-E)	Progress interprets and displays a comma as a decimal separator and a period as a thousands separator for numeric values.
Fractional Separator (-numdec)	Specifies the numeric value of the character that represents, in formatted text, a number's decimal point. The default decimal point is a period (.).
Thousands Separator (-numsep)	Specifies the numeric value of the character that represents, in formatted text, the thousands separator in numbers. The default thousands separator is a comma (,).

NOTE: You can also use parameter files with Progress utilities, for example, PROSHUT and PROUTIL.

C

The Progress Directory Structure

The Progress installation path contains configuration files and several subdirectories. This appendix describes the subdirectories of the Progress installation path.

C.1 Installation Path Directory Structure

The installation path directory contains the Progress executables, several procedure (.p) files, and other related files and subdirectories. During installation, the $DLC environment variable is automatically set to your Progress installation path.

Note that the $DLC variable is set in the various command scripts; the variable is not set at the system level.

Table C–1 describes a directory tree of the Progress subdirectories.

Table C–1: Progress ($DLC) Directory Structure *(1 of 2)*

Directory Name	Description
bin	Contains the executable files for Progress, such as PRODB. It also contains batch files and system executables.
certs	Contains the public keys of the Certificate Authorities (CAs) used by Progress clients to perform server-side certificate validation when communicating with secure Web servers using HTTPS.
gui	Contains object (.r), development, and environment (ADE) files for the Progress graphical tools. These tools are compiled to run in graphical mode on Windows NT, Windows 98, Windows Me, Windows 2000, Windows XP, and Citrix MetaFrame; they cannot run in a character environment.
include	Contains the header files required for ESQL-92.
java	Includes the Java files and executables necessary for running Progress products.
jdk	Contains the Java Development Kit files and executables necessary for running Progress products.
jms	Includes files for the 4GL-to-JMS API of the SonicMQ Adapter.
jre	Contains the Java Run-time Environment files and executables necessary for running Progress products.
lib	Contains shared objects necessary for running Progress executables.
odbc	Includes files to support ODBC.
ora	Contains files to support the DataServer for ORACLE.

Table C–1: Progress ($DLC) Directory Structure *(2 of 2)*

Directory Name	Description
probuild	Includes files that the PROBUILD utility uses when creating custom executables.
proedit	Contains files to support the advanced editing features.
prohelp	Includes the online help and other necessary files for Progress.
prokey32	Contains files for international keyboard support for the 32-bit Windows Character Client.
prolang	Contains the national language support directories.
properties	Contains property files that manage the configuration of Progress services, such as WebSpeed, the NameServer, and the AppServer.
sports	Includes the schema triggers and supplier information for each sample database.
sports2000trgs	Includes the schema triggers for the Sports2000 database.
src	Contains source files for Progress ADE tools, such as the Data Dictionary, Procedure Editor, and Sample Applications.
toolkit	Includes files that help in deploying and encrypting your applications.
tty	Includes mostly object (.r) files for character-mode Progress on Windows NT, Windows 98, Windows Me, Windows 2000, Windows XP, and Citrix MetaFrame.
ubqmanager	Includes files used by the AppServer exclusively. Do not modify these files.
webspeed	Includes static files for the WebSpeed Workshop, WebTools, and HTML help.

D

Progress Version 9 Typical and Complete Installation Components

When you install Progress Version 9.1, you can choose from three installation options: Typical, Complete, or Custom. This appendix provides you with a list of the components and subcomponents that you install for each product when you choose either a Typical or Complete installation.

Specifically, this appendix contains the following sections:

- Progress Version 9 Installation Options
- Progress Product Components and Subcomponents

NOTE: The information in this appendix is generic and attempts to address all UNIX platforms. You might find some minor differences depending on which platform you are installing. In particular, note the availability of a JRE on the following platforms: HPUX 11.00, Solaris Intel, and Solaris SPARC. The Solaris SPARC also includes the JDK with development products only.

D.1 Progress Version 9 Installation Options

You can choose from three options when installing Progress Version 9.1: Typical, Complete, or Custom. The three different installation options allow you to choose the option that is best for you, depending on how many products you are installing, which product components are mandatory and which are optional, and whether all the products reside on the same system.

When you choose the Typical installation option and specify the products you want to install, all mandatory components and subcomponents are installed automatically. Optional components and subcomponents are not included in the installation.

When you choose the Complete installation option and specify the products you want to install, all mandatory and optional components and subcomponents are installed automatically. For this reason, a Complete installation usually requires more disk space than a Typical installation requires.

When you choose the Custom installation option and specify the products you want to install, you can choose the mandatory and optional components and subcomponents to include in the installation. You can also distribute Progress components on different machines and work around disk space limitations. This method is recommended for advanced users.

For a description of the steps to follow when installing Progress Version 9.1, see Chapter 2, "Installing Progress."

The mandatory and optional components and subcomponents for each Progress Version 9 product are listed, by product, in the following tables.

D.2 Progress Product Components and Subcomponents

The following tables list the components and subcomponents that are installed for each product depending on whether you choose the Complete or the Typical installation option.

D.2.1 4GL Development

Table D–1 lists the 4GL Development components and subcomponents. When you choose the Complete installation option and install 4GL Development, all components and subcomponents listed are installed. Choosing the Typical installation option includes all components and subcomponents except those that appear in **bold** text.

Table D–1: 4GL Development Components and Subcomponents *(1 of 4)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
4GL Client	M	Base Client — GL	M
		Character Client	M
ADE Source Code	M	**4GL Profiler**	**O**
		ADE Common Source	M
		ADM Source Code	**O**
		DB Administration Source	**O**
		Demo Applications	**O**
		Editor Source	**O**
AS/400 DataServer Client	**O**	–	–
Character Base Tools	M	**ADM Runtime — CHAR**	**O**
		Base ADE	M
		Compile Tool — CHAR	M
		Procedure Editor — CHAR	M
Client Networking	M	–	–
Common Files	M	–	–
Database Administration Tools	M	4GL Database	M
		Base ADE	M
		Character Administration Tools	M
		Database Utilities	M

Table D–1: 4GL Development Components and Subcomponents *(2 of 4)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
Database Server Component	M	4GL Database	M
		4GL Server	M
		Database Server	M
ODBC DataServer Client	**O**	–	–
Open Interface Driver	**O**	–	–
Oracle DataServer Client	**O**	–	–
ProBuild Utility	**O**	–	–

Table D–1: 4GL Development Components and Subcomponents *(3 of 4)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
Progress Explorer Tools	**M**	**Administration Server**	**M**
		Progress Class Files	**M**
		Ubroker Tools	**M**
Progress Messages (PROMSGS)	M	**(All Languages)**	**O**
Progress SQL ESQL/C Clients	**O**	**ESQL Client**	**M**
		Progress Class Files	**M**
		SQL Common	M
Progress SQL JDBC Clients	**O**	Progress Class Files	M
		SQL Common	M
		SQL JBDC Client	M

Table D–1: 4GL Development Components and Subcomponents *(4 of 4)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
Progress SQL ODBC Clients	**O**	Progress Class Files	**M**
		SQL Common	**M**
		SQL OBDC Client	**M**
SQL Database Server – Development	**O**	**Database Server**	**M**
		Database Utilities	**M**
		Progress Class Files	**M**
		Progress Databases	**M**
		SQL Server	**M**

NOTE: When you choose a Complete installation and install 4GL Development, all components and subcomponents listed in the table are installed. When you choose a Typical installation, all components and subcomponents listed **except** those that appear in **bold** text are installed.

D.2.2 AppServer Internet Adapter

Table D–2 lists the AppServer Internet Adapter components. When you choose the Complete installation option and install the AppServer Internet Adapter, all components listed are installed. Choosing the Typical installation option includes all components and subcomponents except those that appear in **bold** text.

Table D–2: AppServer Internet Adapter Components and Subcomponents

Component	Mandatory /Optional	Subcomponent	Mandatory /Optional
AppServer Internet Adapter	M	–	–
Common Files (minimum)	M	–	–
Progress Messages (PROMSGS)	M	**(All Languages)**	**O**

D.2.3 AppServer Internet Adapter/S

Table D–3 lists the AppServer Internet Adapter/S components. When you choose the Complete installation option and install the AppServer Internet Adapter/S, all components listed are installed. Choosing the Typical installation option includes all components and subcomponents except those that appear in **bold** text.

Table D–3: AppServer Internet Adapter/S Components and Subcomponents

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
AppServer Internet Adapter/S	M	–	–
Common Files (minimum)	M	–	–
Progress Messages (PROMSGS)	M	**(All Languages)**	**O**

D.2.4 Client Networking

Table D–4 lists the Client Networking components and subcomponents. When you choose the Complete installation option and install Client Networking, all components and subcomponents listed are installed. Choosing the Typical installation option includes all components and subcomponents except those that appear in **bold** text.

Table D–4: Client Networking Components and Subcomponents *(1 of 2)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
AS/400 DataServer Client	**O**	–	–
Character Base Tools	M	**ADM Runtime — CHAR**	**O**
		Base ADE	M
		Compile Tool — CHAR	M
		Procedure Editor — CHAR	M
Client Networking	M	–	–
Common Files	M	–	–
Database Administration Tools	**O**	**4GL Database**	**M**
		Base ADE	**M**
		Character Administration Tools	**M**
		Database Utilities	**M**
ODBC DataServer Client	**O**	–	–
Open Interface Driver	**O**	–	–
Oracle DataServer Client	**O**	–	–
ProBuild Utility	**O**	–	–
Progress Messages (PROMSGS)	**M**	**(All Languages)**	**O**
Progress SQL ESQL/C Clients	**O**	**ESQL Client**	**M**
		Progress Class Files	**M**
		SQL Common	**M**

Table D–4: Client Networking Components and Subcomponents *(2 of 2)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
Progress SQL JDBC Clients	**O**	**Progress Class Files**	**M**
		SQL Common	**M**
		SQL JDBC Client	**M**
Progress SQL ODBC Clients	**O**	**Progress Class Files**	**M**
		SQL Common	**M**
		SQL ODBC Client	**M**
Runtime Client	M	Base Client — RT	M
		Character Client	M

NOTE: When you choose a Complete installation and install Client Networking, all components and subcomponents listed in the table are installed. When you choose a Typical installation, all components and subcomponents listed **except** those that appear in **bold** text are installed.

D.2.5 Client-side Security

Table D–5 lists the Client-side Security components. When you choose the Complete installation option and install Client-side Security, all components listed are installed. Choosing the Typical installation option includes all components and subcomponents except those that appear in **bold** text.

Table D–5: Client-side Security Components and Subcomponents

Component	Mandatory /Optional	Subcomponent	Mandatory/ Optional
Secure AppServer Client	M	–	–

D.2.6 Debugger

Table D–6 lists the Debugger components. When you choose the Complete or the Typical installation option and install the Debugger, all components listed are installed.

Table D–6: Debugger Components

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
Common Files	M	–	–
Debugger	M	–	–

D.2.7 Enterprise Database

Table D–7 lists the Enterprise Database components and subcomponents. When you choose the Complete installation option and install the Enterprise Database, all components and subcomponents listed are installed. Choosing the Typical installation option includes all components and subcomponents except those that appear in **bold** text.

Table D–7: Enterprise Database Components and Subcomponents *(1 of 2)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
4GL	**O**	**4GL Server**	**M**
		ADM Runtime — CHAR	**O**
		AS/400 Client	**O**
		Base ADE	**M**
		Base Client — DA	**M**
		Character Administration Tools	**M**
		Character Client	**M**
		Compile Tool — CHAR	**O**
		ODBC DataServer Client	**O**

Table D–7: Enterprise Database Components and Subcomponents *(2 of 2)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
		Open Interface Driver	**O**
		Oracle Client	**O**
		Procedure Editor — CHAR	**M**
		Server Networking	**M**
Bastille Tools	M	Database Server	M
		Database Utilities	M
		Progress Database	M
Common Files	M	–	–
Legacy V8.2 DB Utilities	**O**	–	–
Legacy V8.3 DB Utilities	**O**	–	–
ProBuild Utility	**O**	–	–
Progress Explorer Tools-DB	**M**	Administration Server	M
		Progress Class Files	M
Progress Messages (PROMSGS)	**M**	**(All Languages)**	**O**
SQL	**O**	**ESQL Client**	**M**
		Progress Class Files	**M**
		SQL Common	**M**
		SQL JDBC Client	**M**
		SQL ODBC Client	**M**
		SQL Server	**M**

NOTE: When you choose a Complete installation and install the Enterprise Database, all components and subcomponents listed in the table are installed. When you choose a Typical installation, all components and subcomponents listed **except** those that appear in **bold** text are installed.

D.2.8 ESQL-89/C

Table D–8 lists the ESQL-89/C components. Choosing the Complete or the Typical installation option results in the installation of all components listed.

Table D–8: ESQL-89/C Components

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
Common Files	M	–	–
ESQL/C	M	–	–

D.2.9 NameServer

Table D–9 lists the NameServer components and subcomponents. Choosing the Complete or the Typical Installation option results in the installation of all components and subcomponents listed.

Table D–9: NameServer Components and Subcomponents

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
Common Files	M	–	–
NameServer	M	–	–
Progress Explorer Tools	M	Administration Server	M
		Progress Class Files	M
		Ubroker Tools	M
Progress Messages (PROMSGS)	**M**	**(All Languages)**	**O**

D.2.10 NameServer Load Balancing

Table D–10 lists the NameServer Load Balancing components and subcomponents. When you choose either the Complete or the Typical installation option and install the NameServer Enterprise Edition, all components and subcomponents listed are installed.

Table D–10: NameServer Load Balancing Components and Subcomponents

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
Common Files	M	–	–

D.2.11 Oracle DataServer

Table D–11 lists the Oracle DataServer components and subcomponents. When you choose the Complete installation option and install the Oracle DataServer, all components and subcomponents listed are installed. Choosing the Typical installation option includes all components and subcomponents except those that appear in **bold** text.

Table D–11: Oracle DataServer Components and Subcomponents *(1 of 2)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
AS/400 DataServer Client	**O**	–	–
Character Base Tools	M	**ADM Runtime — CHAR**	**O**
		Base ADE	M
		Compile Tool — CHAR	**M**
		Procedure Editor — CHAR	M
Common Files	M	–	–

Table D–11: Oracle DataServer Components and Subcomponents *(2 of 2)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
Database Administration Tools	M	4GL Database	M
		Base ADE	M
		Character Administration Tools	M
		Database Utilities	M
ODBC DataServer Client	**O**	–	–
Oracle DataServer	M	–	–
Oracle DataServer Client	M	–	–
ProBuild Utility	M	–	–
Progress Explorer Tools	M	Administration Server	M
		Progress Class Files	M
		Ubroker Tools	M
Progress Messages (PROMSGS)	**M**	**(All Languages)**	**O**
Runtime Client	M	Base Client — RT	M
		Character Client	M
Schema Holder Server	M	–	–
Server Networking	M	–	–

When you choose a Complete installation and install the Oracle DataServer, all components and subcomponents listed in the table are installed. When you choose a Typical installation, all components and subcomponents listed **except** those that appear in **bold** text are installed.

D.2.12 Personal Database

Table D–12 lists the Personal Database components and subcomponents. When you choose the Complete installation option and install the Personal Database, all components and subcomponents listed are installed. Choosing the Typical installation option includes all components and subcomponents except those that appear in **bold** text.

Table D–12: Personal Database Components and Subcomponents *(1 of 2)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
4GL	**O**	**4GL Server**	**M**
		ADM Runtime — CHAR	**O**
		AS/400 Client	**O**
		Base ADE	**M**
		Base Client — DA	**M**
		Character Administration Tools	**M**
		Character Client	**M**
		Compile Tool — CHAR	**O**
		ODBC DataServer Client	**O**
		Open Interface Driver	**O**
		Oracle Client	**O**
		Procedure Editor — CHAR	**M**
		Server Networking	**M**
Bastille Tools	M	Database Server	M
		Database Utilities	M
		Progress Database	M
Common Files	M	–	–

Table D–12: Personal Database Components and Subcomponents *(2 of 2)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
Legacy V8.2 DB Utilities	**O**	–	–
Legacy V8.3 DB Utilities	**O**	–	–
ProBuild Utility	**O**	–	–
Progress Explorer Tools-DB	M	Administration Server	M
		Progress Class Files	M
Progress Messages (PROMSGS)	**M**	**(All Languages)**	**O**
SQL	**O**	**ESQL Client**	**M**
		Progress Class Files	**M**
		SQL Common	**M**
		SQL JDBC Client	**M**
		SQL ODBC Client	**M**

NOTE: When you choose a Complete installation and install the Personal Database, all components and subcomponents listed in the table are installed. When you choose a Typical installation, all components and subcomponents listed **except** those that appear in **bold** text are installed.

D.2.13 Progress AppServer

Table D–13 lists the Progress AppServer components and subcomponents. When you choose the Complete installation option and install the AppServer, all components and subcomponents listed are installed. Choosing the Typical installation option includes all components and subcomponents except those that appear in **bold** text.

Table D–13: Progress AppServer Components and Subcomponents *(1 of 2)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
4GL to JMS Adapter	M	Progress Class Files	M
		Sonic MQ Adapter	M
AppServer Internet Adapter	M	–	–
AppServer Runtime Client	M		
AS/400 DataServer Client	**O**	–	–
Client Networking	M	–	–
Common Files	M	–	–
NameServer	M	–	–
ODBC DataServer Client	**O**	–	–
Open AppServer	M	Application Server	M
		Base ADE	M
		Common Broker	M
		Progress Class Files	M
Oracle DataServer Client	**O**	–	–
ProBuild Utility	**O**	–	–
Progress Explorer Tools	M	Administration Server	M
		Progress Class Files	M

Table D–13: Progress AppServer Components and Subcomponents *(2 of 2)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
		Ubroker Tools	M
Progress Messages (PROMSGS)	**M**	**(All Languages)**	**O**
Progress SQL ESQL/C Clients	**O**	**ESQL Client**	**M**
		Progress Class Files	**M**
		SQL Common	**M**
Progress SQL JDBC Clients	**O**	**Progress Class Files**	**M**
		SQL Common	**M**
		SQL JDBC Client	**M**
Progress SQL ODBC Clients	**O**	**Progress Class Files**	**M**
		SQL Common	**M**
		SQL ODBC Client	**M**

NOTE: When you choose a Complete installation and install the AppServer, all components and subcomponents listed in the table are installed. When you choose a Typical installation, all components and subcomponents listed **except** those that appear in **bold** text are installed.

D.2.14 ProVision Development Server

Table D–14 lists the ProVision Development Server components and subcomponents. When you choose the Complete installation option and install the ProVision Development Server, all components and subcomponents listed are installed. Choosing the Typical installation option includes all components and subcomponents except those that appear in **bold** text.

Table D–14: ProVision Development Server Components and Subcomponents *(1 of 3)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
4GL Client	M	Base Client — 4GL	M
		Character Client	M
4GL to JMS Adapter	M	Progress Class Files	M
		Sonic MQ Adapter	M
AppServer Internet Adapter	M	–	–
AS/400 DataServer Client	**O**	–	–
Character Base Tools	M	ADM Runtime — CHAR	O
		Base ADE	M
		Compile Tool — CHAR	M
		Procedure Editor — CHAR	M
Client Networking	M	–	–
Common Files	M	–	–
Database Administration Tools	M	4GL Database	M
		Base ADE	M
		Character Administration Tools	M
		Database Utilities	M
Database Server Component	M	4GL Database	M
		4GL Server	M
		Database Server	M

Table D–14: ProVision Development Server Components and Subcomponents *(2 of 3)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
Debugger	**O**	–	–
NameServer	M	–	–
ODBC DataServer Client	**O**	–	–
Open AppServer	M	Application Server	M
		Base ADE	M
		Common Broker	M
		Progress Class Files	M
Oracle DataServer Client	**O**	–	–
ProBuild Utility	**O**	–	–
Progress Explorer Tools	M	Administration Server	M
		Progress Class Files	M
		Ubroker Tools	M
Progress Messages (PROMSGS)	**M**	**(All Languages)**	**O**
Progress SQL ESQL/C Clients	**O**	**ESQL Client**	**M**
		Progress Class Files	**M**
		SQL Common	**M**
Progress SQL JDBC Clients	**O**	**Progress Class Files**	**M**
		SQL Common	**M**
		SQL JDBC Client	**M**

Table D–14: ProVision Development Server Components and Subcomponents *(3 of 3)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
Progress SQL ODBC Clients	**O**	**Progress Class Files**	**M**
		SQL Common	**M**
		SQL ODBC Client	**M**
SQL Database Server	**O**	**Database Server**	**M**
		Database Utilities	**M**
		Progress Class Files	**M**
		Progress Databases	**M**
		SQL Server	**M**

NOTE: When you choose a Complete installation and install the ProVision Development Server, all components and subcomponents listed in the table are installed. When you choose a Typical installation, all components and subcomponents listed **except** those that appear in **bold** text are installed.

D.2.15 ProVision Plus Development Server

Table D–15 lists the ProVision Plus Development Server components and subcomponents. When you choose the Complete installation option and install the ProVision Plus Development Server, all components and subcomponents listed are installed. Choosing the Typical installation option includes all components and subcomponents except those that appear in **bold** text.

Table D–15: ProVision Plus Development Server Components and Subcomponents *(1 of 4)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
4GL Client	M	Base Client — 4GL	M
		Character Client	M
4GL to JMS Adapter	M	Progress Class Files	M
		Sonic MQ Adapter	M
AppServer Internet Adapter	M	–	–
AS/400 DataServer Client	**O**	–	–
Character Base Tools	M	**ADM Runtime — CHAR**	**O**
		Base ADE	M
		Compile Tool — CHAR	**M**
		Procedure Editor — CHAR	M
Client Networking	M	–	–
Common Files	M	–	–

Table D–15: ProVision Plus Development Server Components and Subcomponents *(2 of 4)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
Database Administration Tools	M	4GL Database	M
		Base ADE	M
		Character Administration Tools	M
		Database Utilities	M
Database Server Component	M	4GL Database	M
		4GL Server	M
		Database Server	M
Debugger	**O**	–	–
Messenger Component	M	–	–
NameServer	M	–	–
ODBC DataServer Client	**O**	–	–
Open AppServer	M	Application Server	M
		Base ADE	M
		Common Broker	M
		Progress Class Files	M
Oracle DataServer Client	**O**	–	–
ProBuild Utility	**O**	–	–
Progress Explorer Tools	M	Administration Server	M
		Progress Class Files	M
		Ubroker Tools	M

Table D–15: ProVision Plus Development Server Components and Subcomponents *(3 of 4)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
Progress Messages (PROMSGS)	**M**	**(All Languages)**	**O**
Progress SQL ESQL/C Clients	**O**	**ESQL Client**	**M**
		Progress Class Files	**M**
		SQL Common	**M**
Progress SQL JDBC Clients	**O**	**Progress Class Files**	**M**
		SQL Common	**M**
		SQL JDBC Client	**M**
Progress SQL ODBC Clients	**O**	**Progress Class Files**	**M**
		SQL Common	**M**
		SQL ODBC Client	**M**
SQL Database Server	**O**	**Database Server**	**M**
		Database Utilities	**M**
		Progress Class Files	**M**
		Progress Databases	**M**
		SQL Server	**M**
Transaction Server	M	Progress Databases	M
		Web Static	M

Table D–15: ProVision Plus Development Server Components and Subcomponents *(4 of 4)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
		WebSpeed Runtime	M
		WebSpeed Server Development	M

NOTE: When you choose a Complete installation and install ProVision Plus Development Server, all components and subcomponents listed in the table are installed. When you choose a Typical installation, all components and subcomponents listed **except** those that appear in **bold** text are installed.

D.2.16 Query/Results

Table D–16 lists the Query/Results components and subcomponents. When you choose the Complete installation option and install Query/Results, all components and subcomponents listed are installed. Choosing the Typical installation option includes all components and subcomponents except those that appear in **bold** text.

Table D–16: Query/Results Components and Subcomponents *(1 of 2)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
AS/400 DataServer Client	**O**	–	–
Character Base Tools	M	**ADM Run-time — CHAR**	**O**
		Base ADE	M
		Compile Tool — CHAR	**M**
		Procedure Editor — CHAR	M
Common Files	M	–	–

Table D–16: Query/Results Components and Subcomponents *(2 of 2)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
Database Administration Tools	**O**	**4GL Database**	**M**
		Base ADE	**M**
		Character Administration Tools	**M**
		Database Utilities	**M**
ODBC DataServer Client	**O**	–	–
Oracle DataServer Client	**O**	–	–
ProBuild Utility	**O**	–	–
Progress Messages (PROMSGS)	**M**	**(All Languages)**	**O**
Query Client	M	Base Client — Query	M
		Character Client	M
Results	M	Base ADE	M
		Results (Char)	M

NOTE: When you choose a Complete installation and install Query/Results, all components and subcomponents listed in the table are installed. When you choose a Typical installation, all components and subcomponents listed **except** those that appear in **bold** text are installed.

D.2.17 Secure AppServer

Table D–17 lists the Secure AppServer components and subcomponents. Choosing the Complete option results in the installation of all components and subcomponents listed. Choosing the Typical installation option includes all components and subcomponents except those that appear in **bold** text.

Table D–17: Secure AppServer Components and Subcomponents *(1 of 2)*

Component	Mandatory /Optional	Subcomponent	Mandatory /Optional
4GL to JMS Adapter	M	Progress Class Files	M
		Sonic MQ Adapter	M
AppServer Internet Adapter/S	M	–	–
AppServer Runtime Client	**M**		
AS/400 DataServer Client	**O**	–	–
Client Networking	M	–	–
Common Files	M	–	–
NameServer	M	–	–
ODBC DataServer Client	**O**	–	–
Open AppServer	M	Application Server	M
		Base ADE	M
		Common Broker	M
		Progress Class Files	M
Oracle DataServer Client	**O**	–	–
ProBuild Utility	**O**	–	–
Progress Explorer Tools	M	Administration Server	M
		Progress Class Files	M
		Ubroker Tools	M
Progress Messages	M	**(All international PROMSGS)**	**O**

Table D–17: Secure AppServer Components and Subcomponents *(2 of 2)*

Component	Mandatory /Optional	Subcomponent	Mandatory /Optional
Progress SQL ESQL/C Clients	**O**	**ESQL Client**	**M**
		Progress Class Files	**M**
		SQL Common	**M**
Progress SQL JDBC Clients	**O**	**Progress Class Files**	**M**
		SQL Common	**M**
		SQL JDBC Client	**M**
Progress SQL ODBC Clients	**O**	**Progress Class Files**	**M**
		SQL Common	**M**
		SQL ODBC Client	**M**

D.2.18 SonicMQ Adapter

Table D–18 lists the SonicMQ Adapter components and subcomponents. Choosing the Complete option results in the installation of all components and subcomponents listed. Choosing the Typical installation option includes all components and subcomponents except those that appear in **bold** text.

Table D–18: SonicMQ Adapter Components and Subcomponents *(1 of 2)*

Component	Mandatory /Optional	Subcomponent	Mandatory /Optional
4GL to JMS Adapter	M	Progress Class Files	M
		SonicMQ Adapter	M
Common Files	M	–	–
NameServer	M		

Table D–18: SonicMQ Adapter Components and Subcomponents *(2 of 2)*

Component	Mandatory /Optional	Subcomponent	Mandatory /Optional
Progress Explorer Tools	M	Administration Server	M
		Progress Class Files	M
		Ubroker Tools	M
Progress Messages	M	**(All international PROMSGS)**	**O**

D.2.19 SQL-92 Client Access

Table D–19 lists the SQL-92 Client Access components and subcomponents. When you choose the Complete or the Typical installation option and install SQL-92 Client Access, all components and subcomponents listed are installed.

Table D–19: SQL-92 Client Access Components and Subcomponents

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
Common Files	M	–	–
Progress Explorer Tools-SQL	M	Progress Class Files	M
Progress Messages (PROMSGS)	M	**(All Languages)**	**O**
Progress SQL ESQL/C Clients	M	ESQL Client	M
		Progress Class Files	M
		SQL Common	M
Progress SQL JDBC Clients	M	Progress Class Files	M
		SQL Common	M
		SQL JDBC Client	M
Progress SQL ODBC Clients	M	Progress Class Files	M
		SQL Common	M
		SQL ODBC Client	M

D.2.20 Toolkit

Table D–20 lists the Toolkit components. Choosing the Complete or the Typical installation option results in the installation of all components.

Table D–20: Toolkit Components

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
Common Files	M	–	–
Toolkit	M	–	–

D.2.21 WebSpeed Development Server

Table D–21 lists the WebSpeed Development Server components and subcomponents. When you choose the Complete installation option and install the WebSpeed Development Server, all components and subcomponents listed are installed. Choosing the Typical installation option includes all components and subcomponents except those that appear in **bold** text.

Table D–21: WebSpeed Development Server Components and Subcomponents *(1 of 4)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
4GL Client	M	Base Client — 4GL	M
		Character Client	M
4GL to JMS Adapter	M	Progress Class Files	M
		Sonic MQ Adapter	M
AS/400 DataServer Client	**O**	–	–
Client Networking	M	–	–
Common Files	M	–	–

Table D–21: WebSpeed Development Server Components and Subcomponents *(2 of 4)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
Database Administration Tools	M	4GL Database	M
		Base ADE	M
		Character Administration Tools	M
		Database Utilities	M
Database Server Component	M	4GL Database	M
		4GL Server	M
		Database Server	M
Debugger	**O**	–	–
Messenger Component	M	–	–
NameServer	M	–	–
ODBC DataServer Client	**O**	–	–
Open AppServer	M	Application Server	M
		Base ADE	M
		Common Broker	M
		Progress Class Files	M
Oracle DataServer Client	**O**	–	–
ProBuild Utility	**O**	–	–
Progress Explorer Tools	M	Administration Server	M
		Progress Class Files	M
		Ubroker Tools	M

Table D–21: WebSpeed Development Server Components and Subcomponents *(3 of 4)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
Progress Messages (PROMSGS)	M	**(All languages)**	**O**
Progress SQL ESQL/C Clients	**O**	**ESQL Client**	**M**
		Progress Class Files	**M**
		SQL Common	**M**
Progress SQL JDBC Clients	**O**	**Progress Class Files**	**M**
		SQL Common	**M**
		SQL JDBC Client	**M**
Progress SQL ODBC Clients	**O**	**Progress Class Files**	**M**
		SQL Common	**M**
		SQL ODBC Client	**M**
SQL Database Server	**O**	**Database Server**	**M**
		Database Utilities	**M**
		Progress Class Files	**M**
		Progress Databases	**M**
		SQL Server	**M**
Transaction Server	M	Progress Databases	M
		Web Static	M

Table D–21: WebSpeed Development Server Components and Subcomponents *(4 of 4)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
		WebSpeed Runtime	M
		WebSpeed Server Development	M

NOTE: When you choose a Complete installation and install the WebSpeed Development Server, all components and subcomponents listed in the table are installed. When you choose a Typical installation, all components and subcomponents listed **except** those that appear in **bold** text are installed.

D.2.22 WebSpeed Enterprise Transaction Server

Table D–22 lists the WebSpeed Enterprise Transaction Server components and subcomponents. When you choose the Complete installation option and install the WebSpeed Enterprise Transaction Server, all components and subcomponents listed are installed. Choosing the Typical installation option includes all components and subcomponents except those that appear in **bold** text.

Table D–22: WebSpeed Enterprise Transaction Server Components and Subcomponents *(1 of 3)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
4GL Client	M	Base Client — GL	M
		Character Client	M
4GL to JMS Adapter	M	Progress Class Files	M
		Sonic MQ Adapter	M
ADE Source Code	**O**	**4GL Profiler**	**O**
		ADE Common Source	**O**
		Demo Applications	**O**
		Editor Source	**O**

Table D–22: WebSpeed Enterprise Transaction Server Components and Subcomponents *(2 of 3)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
AS/400 DataServer Client	**O**	–	–
Client Networking	M	–	–
Common Files	M	–	–
Messenger Component	M	–	–
NameServer	M	–	–
ODBC DataServer Client	**O**	–	–
Open AppServer	M	Application Server	M
		Base ADE	M
		Common Broker	M
		Progress Class Files	M
Oracle DataServer Client	**O**	–	–
ProBuild Utility	**O**	–	–
Progress Explorer Tools	M	Administration Server	M
		Progress Class Files	M
		Ubroker Tools	M
Progress Messages (PROMSGS)	**M**	**(All Languages)**	**O**
Progress SQL ESQL/C Clients	**O**	**ESQL Client**	**M**
		Progress Class Files	**M**
		SQL Common	**M**

Table D–22: WebSpeed Enterprise Transaction Server Components and Subcomponents *(3 of 3)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
Progress SQL JDBC Clients	**O**	**Progress Class Files**	**M**
		SQL Common	**M**
		SQL JDBC Client	**M**
Progress SQL ODBC Clients	**O**	**Progress Class Files**	**M**
		SQL Common	**M**
		SQL ODBC Client	**M**
Transaction Server	M	Progress Databases	M
		Web Static	M
		WebSpeed Runtime	M
		WebSpeed Server Development	M

NOTE: When you choose a Complete installation and install the WebSpeed Enterprise Transaction Server, all components and subcomponents listed in the table are installed. When you choose a Typical installation, all components and subcomponents listed **except** those that appear in **bold** text are installed.

D.2.23 WebSpeed Messenger

Table D–23 lists the WebSpeed Messenger components. Choosing the Complete or the Typical option results in the installation of all components listed.

Table D–23: WebSpeed Messenger Components

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
Messenger Component	M	–	–
Progress Messages (PROMSGS)	M	(All languages)	O
Web Static	M	–	–
WebSpeed Common	M	–	–

D.2.24 WebSpeed Transaction Server

Table D–24 lists the WebSpeed Transaction Server components and subcomponents. When you choose the Complete installation option and install the WebSpeed Transaction Server, all components and subcomponents listed are installed. Choosing the Typical installation option includes all components and subcomponents except those that appear in **bold** text.

Table D–24: WebSpeed Transaction Server Components and Subcomponents *(1 of 3)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
4GL Client	M	Base Client — 4GL	M
		Character Client	M
4GL to JMS Adapter	M	Progress Class Files	M
		Sonic MQ Adapter	M

Table D–24: WebSpeed Transaction Server Components and Subcomponents *(2 of 3)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
ADE Source Code	**O**	**4GL Profiler**	**O**
		ADE Common Source	**O**
		Demo Applications	**O**
		Editor Source	**O**
AS/400 DataServer Client	**O**	–	–
Client Networking	M	–	–
Common Files	M	–	–
Messenger Component	M	–	–
NameServer	M	–	–
ODBC DataServer Client	**O**	–	–
Open AppServer	M	Application Server	M
		Base ADE	M
		Common Broker	M
		Progress Class Files	M
Oracle DataServer Client	**O**	–	–
ProBuild Utility	**O**	–	–
Progress Explorer Tools	M	Administration Server	M
		Progress Class Files	M
		Ubroker Tools	M
Progress Messages (PROMSGS)	M	**(All languages)**	**O**

Table D–24: WebSpeed Transaction Server Components and Subcomponents *(3 of 3)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
Progress SQL ESQL/C Clients	**O**	**ESQL Client**	**M**
		Progress Class Files	**M**
		SQL Common	**M**
Progress SQL JDBC Clients	**O**	**Progress Class Files**	**M**
		SQL Common	**M**
		SQL JDBC Client	**M**
Progress SQL ODBC Clients	**O**	**Progress Class Files**	**M**
		SQL Common	**M**
		SQL ODBC Client	**M**
Transaction Server	M	Progress Databases	M
		Web Static	M
		WebSpeed Runtime	M
		WebSpeed Server Development	M

NOTE: When you choose a Complete installation and install the WebSpeed Transaction Server, all components and subcomponents listed in the table are installed. When you choose a Typical installation, all components and subcomponents listed **except** those that appear in **bold** text are installed.

D.2.25 Workgroup Database

Table D–25 lists the Workgroup Database components and subcomponents. When you choose the Complete installation option and install the Workgroup Database, all components and subcomponents listed are installed. Choosing the Typical installation option includes all components and subcomponents except those that appear in **bold** text.

Table D–25: Workgroup Database Components and Subcomponents *(1 of 2)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
4GL	**O**	**4GL Server**	**M**
		ADM Run-time — CHAR	**O**
		AS/400 Client	**O**
		Base ADE	**M**
		Base Client — DA	**M**
		Character Administration Tools	**M**
		Character Client	**M**
		Compile Tool — CHAR	**O**
		ODBC DataServer Client	**O**
		Open Interface Driver	**O**
		Oracle Client	**O**
		Procedure Editor — CHAR	**M**
		Server Networking	**M**
Bastille Tools	M	Database Server	M
		Database Utilities	M
		Progress Database	M
Common Files	M	–	–
Legacy V8.2 DB Utilities	**O**	–	–
Legacy V8.3 DB Utilities	**O**	–	–
ProBuild Utility	**O**	–	–

Table D–25: Workgroup Database Components and Subcomponents *(2 of 2)*

Component	Mandatory/ Optional	Subcomponent	Mandatory/ Optional
Progress Explorer Tools — DB	M	Administration Server	M
		Progress Class Files	M
Progress Messages (PROMSGS)	M	**(All languages)**	**O**
SQL	**O**	**ESQL Client**	**M**
		Progress Class Files	**M**
		SQL Common	**M**
		SQL JDBC Client	**M**
		SQL ODBC Client	**M**
		SQL Server	**M**

NOTE: When you choose a Complete installation and install the Workgroup Database, all components and subcomponents listed in the table are installed. When you choose a Typical installation, all components and subcomponents listed **except** those that appear in **bold** text are installed.

Index

Numbers

4GL Development
components installed during Typical or Complete installation D–2 to D–6

A

Access rights, networks 7–28

ADAPTCONFIG utility 8–12

ADAPTMAN utility 8–6

Addressing
Progress brokers/servers 10–9
Progress hosts 10–9

AdminServer 7–2

After-image writers (AIW) 7–12

AIACONFIG utility 8–12

AIAMAN utility 8–6

Application services 8–9
default service 8–10

Application workstations, defined 7–13

Applications, starting on networks 10–9

AppServer Internet Adapter
components installed during Typical or Complete installation D–6

AppServer Internet Adapter/S
components installed during Typical or Complete installation D–7

Arbitrary sum weight factors 8–29

AS/400, default network type 10–8

ASBMAN utility 8–6

ASCONFIG utility 8–12

Asynchronous page writers (APWs) 7–12

B

Background mode
multi-user Progress 10–13
single-user Progress 10–5

Batch mode
multi-user Progress 10–13
single-user Progress 10–5

Batch startup commands
single-user 10–3

Before-image Filename (-g) startup parameter 7–24

Before-image writers (BIW) 7–12

BPRO utility 10–5

Broadcast request to NameServer 8–33

Brokers
  addressing 10–9
  multi-broker access 10–10
  starting 10–9
    servers for remote users 10–6

C

Client Networking, components installed during Typical or Complete installation D–7, D–9

Client startup parameters, networks 10–7

Client/server configurations 7–13
  samples 7–15
  terminology 7–13

Client/server Progress
  addressing 10–9
  broker/server addressing 10–9
  execution 10–6
  starting clients and servers 10–6

Clients
  defined 7–13
  starting, TCP/IP on UNIX 10–9

Client-side Security
  components installed during Typical or Complete installation D–9

Commands, mount 2–7, 2–8

Compaq Tru64
  Java requirements for 1–3, 5–7
  preinstallation tasks for 2–2

Compatibility
  4GL clients and database servers 1–8

Complete installation 2–5, 2–14 to 2–21

Components
  installed during a Typical or Complete installation
    4GL D–2 to D–6
    AppServer Internet Adapter D–6
    AppServer Internet Adapter/S D–7
    Client Networking D–7, D–9
    Client-side Security D–9
    Debugger D–9, D–10
    Enterprise Database D–10 to D–12
    ESQL 89/C D–12
    NameServer D–12
    NameServer Enterprise Edition D–13
    Oracle DataServer D–13, D–14
    Personal Database D–15
    Progress AppServer D–17 to D–18
    ProVISION Development Server D–18 to D–21
    ProVISION Plus Development Server D–22 to D–25
    Query/RESULTS D–25, D–26
    Secure AppServer D–26 to D–28
    SonicMQ Adapter D–28
    SQL-92 Client Access D–30
    Toolkit D–31
    WebSpeed Development Server D–31 to D–34
    WebSpeed Enterprise Transaction Server D–34 to D–36
    WebSpeed Messenger D–37
    WebSpeed Transaction Server D–37 to D–39
    Workgroup Database D–39, D–41
  used to calculate memory needs 6–2

Configuration utilities
  ADAPTCONFIG 7–2
  ADAPTMAN 7–2
  AIACONFIG 7–2
  ASBMAN 7–3
  ASCONFIG 7–3
  DBMAN 7–3
  MSSCONFIG 7–3
  MSSMAN 7–3
  NSCONFIG 7–3
  NSMAN 7–3
  ODBCONFIG 7–3
  ODBMAN 7–3
  ORACONFIG 7–3
  ORAMAN 7–3

PROADSV 7–3
WSCONFIG 7–3
WTBMAN 7–3

Configuration, displaying information 3–3

Configurations
client/server 7–13
displaying information 3–5
network 7–16
shared memory 7–7

Configuring Progress on an NOS 7–26

Configuring Unified Broker components 8–18

Configuring Unified Broker products 8–1

Connection-level fault tolerance 8–8
two techniques 8–32
understanding 8–31

Control number, displaying 3–4

Controlling NameServer 8–7

Custom installation 2–5, 2–21 to 2–30

D

Database
sample application C–3

Database server machines
defined 7–13
multi-process, defined 7–14
single-process, defined 7–14

Databases
monitoring 7–12
startup 10–9

DBMAN 7–3

Debugger, components installed during Typical or Complete installation D–10

Default service 8–10

Directory structure
Progress C–1

Disk space requirements 1–6

Distributing processes, Unified Broker 8–16

DLC
directory structure C–2
environment variable, UNIX 5–3

Downloading executables for heterogeneous environments 2–35

E

Editing unbroker.propertes 8–20

Enterprise Database, components installed during Typical or Complete installation D–10 to D–12

Environment variables
PATH, UNIX 5–3
setting 5–2
for Java 5–7
Unified Broker settings 8–23
UNIX 5–2, 5–3

Error code parameters 6–10

Error messages
related to installation 4–2

ESQL/C, components installed during Typical or Complete installation D–12

European Numeric Format (-E) B–7

Expiration date, displaying 3–4

F

Fail-over weight factor 8–30

Files
  log, configuring 8–24
  network sharing 7–28
  protocols. *See* protocols file
  services. *See* services file
  ubroker.properties 8–10

Formulas, for calculating memory requirements 6–5

H

-H startup parameter
  client connection 10–7
  server startup 10–7
  specifying 10–9

Host addressing 10–9

Host Name (-H) parameter
  client connection 10–7
  server startup 10–7
  specifying 10–9

Host request to NameServer 8–33

HP-UX
  Java requirements for 1–3, 5–7
  modifying JDKHOME value for 5–8
  setting SQL environment variables for 5–8

I

IBM AIX
  Java requirements for 1–3, 5–7
  setting SQL environment variables for 5–8

Installation
  date, displaying 3–4
  displaying configuration information 3–3, 3–5
  options 2–5

Installation path (DLC)
  directory structure C–2

Installing Progress
  and performing post-installation tasks 2–34
  as a Complete installation 2–5, 2–14 to 2–21
  as a Custom installation 2–5, 2–21 to 2–30
  as a Typical installation 2–5, 2–14 to 2–21
  before you begin 2–2, 2–6
  choosing an installation type when 2–5, D–2
  components, in a Typical or Complete installation D–1 to D–41
  in a heterogeneous environment 2–35
  products, to an existing installation 2–36
  troubleshooting when 4–1 to 4–4

Interactive mode, multi-user Progress 10–12

International
  databases B–4
  parameter files B–6

J

JavaHome installation directory 2–12

JDKHOME, setting 5–8

K

Kernel configuration 6–8

Kernel reconfiguration parameters 6–10

L

LAN Manager, resource sharing 7–28

LAN Progress, startup parameters 10–6

Language, support B–1

License
information 1–9
updating 3–2

Licensed user counts 3–5

Licenses 3–5

Load balancing 8–29, 8–30

Local NameServer instances 8–19

Location transparency 8–7

Log (.lg) files, commands to remove entries 3–8

Log files, Unified Broker 8–24

M

-m2 startup parameter, server startup 10–7

-m3 startup parameter, server startup 10–7

-Ma startup parameter, server startup 10–7

Machine addressing 10–9

Machine distribution, Unified Broker products 8–16

Maintaining two versions of Progress 9–2

Manual Server (-m2) startup parameter 10–7

Maximum Clients per Server (-Ma) parameter, server startup 10–7

Maximum Dynamic Server (-Maxport) parameter, server startup 10–7

Maximum Servers (-Mn) parameter, server startup 10–8

Maximum Servers per Broker (-Mpb) startup parameter, server startup 10–8

Maximum Servers per Protocol (-Mp) parameter, server startup 10–8

-Maxport startup parameter, server startup 10–7

MBPRO command 10–13

Memory
and semaphore parameter settings 6–9, 6–10
and setting swap space 6–8
calculating requirements for 6–2
formulas to calculate requirements for 6–5
multi-user requirements for 6–4
reducing usage 6–7
shared 6–8
parameter settings 6–9, 6–10
single-user requirements for 6–4

Message Buffer Size (-Mm) parameter, client connection 10–7

-Mi startup parameter, server startup 10–7

Minimum Clients per Server (-Ma) parameter, server startup 10–7

Minimum Dynamic Server (-Minport) parameter, server startup 10–8

Minimum operating system level, by platform 1–9

-Minport startup parameter, server startup 10–8

-Mm startup parameter, server startup 10–7

-Mn startup parameter, server startup 10–8

Mount commands 2–7, 2–8

-Mp startup parameter, server startup 10–8

-Mpb startup parameter, server startup 10–8

MPRO command 10–9

MPRO utility 10–12

MS-Windows, TCP/IP 7–23

Multi-threaded servers 7–13

Multi-user memory requirements 6–4

Multi-user Progress
  commands to start
    batch mode 10–13
    interactive mode 10–12
  starting the server 10–6
  startup commands
    batch mode 10–13
    broker 10–6
    interactive mode 10–12
    server 10–6

N

-N startup parameter 10–7, 10–8

NameServer
  accessing its port and host 8–17
  broadcast address 8–35
  client connections 8–20
  components installed during Typical or Complete installation D–12
  configuring communications 8–17
  editing the services file 8–18
  features 8–7
  instance configuration 8–19
  log file 8–24
  neighbors 8–32
    using 8–36
  replication 8–32
    example 8–34
    using 8–33
  starting instances, using NSMAN utility 8–26
  UDP broadcasting 8–33

NameServer Enterprise Edition, components installed during Typical or Complete installation D–13

NameServers, downloading executables 8–19

Neighbor NameServers 8–32

Network configurations 7–16

Network file servers
  applications 7–21
  defined 7–13

Network operating system
  access rights 7–28
  accessing resources 7–26
    Novell NetWare 7–28
  Progress startup 10–6
  resource sharing 7–28
    LAN Manager 7–28

Network startup parameters 10–6

Network Type (-N) parameter
  client connection 10–7
  defaults 10–8
  server startup 10–8
  specifying 10–8

Network Type (-N) startup parameter, protocol support 7–17

Network types, defined 7–17

Networks
  access rights 7–28
  accessing resources 7–26
    Novell NetWare 7–28
  applications, starting 10–9
  client startup parameters 10–7
  client/server, startup 10–9
  database server machines 7–13
  file servers 7–13
  PC LAN permissions 7–28
  Progress startup parameters 10–6
  resource sharing 7–28
    LAN Manager 7–28
  server startup parameters 10–7
  types 7–17

Novell NetWare, resource sharing 7–28

NSCONFIG utility 8–12
  syntax 8–21
  using 8–21

NSMAN utility 8–6
  syntax 8–27
  using 8–26

NT, default network type 10–8

O

ODBMAN utility 8–6

Open Client Toolkit
  and the Just-in-Time Compiler 5–9
  preinstallation tasks for 2–3

Oracle DataServer, components installed during Typical or Complete installation D–13, D–14

ORACONFIG utility 8–12

ORAMAN utility 8–6

P

Parameter file, typical startup parameters B–6, B–7

PATH, environment variable, UNIX 5–3

Percentage-based weight factors 8–29

Permissions, networks 7–28

Personal Database, components installed during Typical or Complete installation D–15

Platforms, supported 1–9

Post-installation tasks 2–34

Preinstallation tasks
  for Compaq Tru64 2–2
  for Open Client Toolkit 2–3
  for WebSpeed 2–2
  reading release notes as part of 2–2
  to save an existing Progress installation 2–3

Priority weight factor, load balancing. *See* Weight factors

priorityWeight property 8–29

PRO utility 10–4

proadsv 5–11

Procedure Editor startup commands, UNIX, Windows 10–3

Process distribution, Unified Broker 8–16

PROCFG, environment variable, UNIX 5–4

PROCONV, environment variable, UNIX 5–4

Product configurations, ubroker.properties file 7–4

Product Life Cycle 1–10, 1–12

Product name, displaying 3–4

Product update utility (PROUPDT) 3–1, 3–2

PROEXE, environment variable, UNIX 5–4

Progress
  broker/server addressing 10–9
  client/server startup 10–6
  configuration file (Progress.cfg) 4–2
  configuring on an NOS 7–26
  directory structure C–1
  maintaining two versions of 9–2
  messages B–4
  network addressing 10–9
  network startup parameters 10–6
  Product Life Cycle 1–10, 1–12
  product update utility 3–1, 3–2
  running 5–9
  starting multi-user
    batch mode 10–13
    interactive mode 10–12
  starting single-user
    batch mode 10–4
    interactive mode 10–4
  supported platforms 1–9

Progress AppServer, components installed during Typical or Complete installation D–17

Progress AppServer, configuring 8–1

Progress client/server
network file server, application files 7–21
running on a LAN 10–6
sample configurations 7–17
startup parameters 10–6
with Progress AppServer 7–22

Progress configuration file (Progress.cfg), altered or missing 4–2

Progress DataServers, configuring with NameServer 8–1

Progress Explorer, framework 7–2

Progress installation, TCP/IP networks 7–24

Progress network types
SNA 7–17
TCP 7–17

Progress product, displaying configuration information 3–3

Progress sample configurations 7–17
network file servers
application files 7–21
with Progress AppServer 7–22

Progress Unified Broker. *See* Unified Broker

Prolang directory B–3

PROLOAD, environment variable, UNIX 5–4

PROLOG, removing log entries 3–8

PROMON utility 7–12

PROMSGS
environment variable, UNIX 5–4
translations B–2

PROPATH, environment variable, UNIX 5–5

Properties file
ubroker.properties 7–4
Unified Broker 8–10
editing and validating 8–20

PROSERVE
command 10–9
utility 10–6, 10–10

PROSRV, environment variable, UNIX 5–6

PROSTART, environment variable, UNIX 5–6

PROTERMCAP, environment variable, UNIX 5–6

PROUPDT utility. *See* Product update utility

ProVISION Development Server, components installed during Typical or Complete installation D–18 to D–21

ProVISION Plus Development Server, components installed during Typical or Complete installation D–22 to D–25

PROWDOG utility 7–12

Q

Query/RESULTS, components installed during Typical or Complete installation D–25, D–26

R

Regional support, installing PROMSGS B–3

Remote NameServer instances 8–19

Removing log file entries, PROLOG 3–8

Replicated NameServers 8–32

Resources, network sharing 7–28

Running Progress 5–9

S

-S startup parameter 10–8
  client connection 10–7
  server startup 10–9

Sample application database C–3

Saving an existing Progress installation 2–4

Secondary Login Broker (-m3) startup parameter, server startup 10–7

Secure AppServer
  components installed during Typical or Complete installation D–26 to D–28

Semaphore parameter settings 6–9, 6–10

Serial number, displaying 3–4

Server startup parameters, networks 10–7

Server-level fault tolerance 8–7
  understanding 8–31

Servers
  addressing 10–9
  defined 7–13
  network 7–13
  starting 10–9

Service Name (-S) startup parameter 10–9
  client connection 10–7
  server startup 10–8

services file 8–18

Setting environment variables 5–2
  for Java 5–7
  for SQL 5–8

Shared memory 6–8
  architecture 7–11
  configurations 7–7
  defined 7–7
  parameter settings 6–9, 6–10

Show Configuration (SHOWCFG) utility 3–5

SHOWCFG utility 3–3

Single-threaded servers, defined 7–13

Single-user memory requirements 6–4

Single-user Progress, startup commands
  batch mode 10–4
  interactive mode 10–5

Size increments for increasing startup parameters 6–3

SNA network types 7–17

Solaris, setting SQL environment variables for 5–8

SonicMQ Adapter
  components installed during Typical or Complete installation D–28

SQL-92, system requirements 1–4, 2–3

Starting
  clients and servers 10–6
  network applications 10–9

Starting Progress
  multi-user
    batch mode 10–12
    interactive mode 10–12
    with a server 10–6
  single-user
    batch mode 10–4
    interactive mode 10–4

Starting Unified Broker instances 8–25

Startup commands, summary 10–2

Startup parameters
client 10–7
server 10–7
size increments for increasing 6–3

Startup scripts, tailoring 4–3, 4–4

Supported, platforms 1–9

Swap space 6–8

System requirements
disk space 1–6
for building Version 9 applications 1–6
for running Version 9 applications 1–6
Java 1–2
Compaq Tru64 1–3
HP-UX 1–3
IBM AIX 1–3
SQL-92 1–4, 2–3

T

Tailoring startup scripts 4–3

TCP network types 7–17

TCP/IP networks
configuring UNIX machines 7–23
installing Progress 7–24
network files 7–26
NT, software requirements 7–23
preparing for Progress 7–26
without file server 7–25

TERM, environment variable, UNIX 5–6, 5–7

Terminal identifiers A–2, A–3

Toolkit, components installed during Typical or Complete installation D–31

Troubleshooting an installation 4–1 to 4–4

Types of Unified Brokers 8–9

Typical installation 2–5, 2–14 to 2–21

U

ubroker.properties file 7–4, 8–10
editing. *See* Properties file
hierarchy 7–4
products supported 7–5
validating. *See*Properties file

UDP broadcasting 8–33
performance implications 8–39

Unified Broker
administration overview 8–13
administration task summary 8–3
application services 8–9
clients 8–4
command-line configuration 8–11
command-line management 8–6
configuring the components 8–18
environment setup 8–23
log files 8–24
machine and process distribution 8–16
NameServer instances 8–20
order of configuration 8–19
product features 8–8
properties file 8–10
guidelines for editing 8–11
validating 8–12
setting environment variables 8–23
setting the working directory 8–23
starting instances, using management utilities 8–25
types 8–9

UNIX
default network type 10–8
environment variables 5–2, 5–3 to 5–7
starting network clients 10–9

Updating
license information 3–2
number of users 3–2

User licenses. *See* Licenses

Utilities
BPRO 10–5
MBPRO 10–13
MPRO 10–12
PRO 10–4
PROMON 7–12
PROSERVE 10–6, 10–10
PROWDOG 7–12
Watchdog 7–12

V

Version number, displaying 3–4

W

Watchdog utility 7–12

WebSpeed Development Server, components installed during Typical or Complete installation D–31 to D–34

WebSpeed Enterprise Transaction Server, components installed during Typical or Complete installation D–34 to D–36

WebSpeed Messenger, components installed during Typical or Complete installation D–37

WebSpeed Transaction Server configuring 8–1

WebSpeed Transaction Server, components installed during Typical or Complete installation D–37 to D–39

WebSpeed, preinstallation tasks for 2–2

Weight factors 8–29

Workgroup Database, components installed during Typical or Complete installation D–39

Working directory, Unified Broker setting 8–23

WSCONFIG utility 8–12

WTBMAN utility 8–6