

Progress
Startup Command and Parameter Reference

© 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: 4600
Item Number: 91585;V9.1D

Contents

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

1. Introduction **1–1**
1.1 About the Startup Commands 1–2
1.2 About the Startup Parameters 1–3
1.3 Using Parameter Files 1–3
1.3.1 Calling Parameter Files 1–4
1.3.2 Creating and Editing Parameter Files 1–5
1.3.3 Parameter File Format 1–6

2. Startup and Shutdown Commands 2–1
2.1 Startup Command Syntax 2–2
2.2 Tasks You Can Perform With Startup and Shutdown Commands 2–3
2.3 Startup and Shutdown Command Descriptions 2–5
BPRO Command 2–6
MBPRO Command 2–8
MPRO Command 2–9
PRO Command 2–10
PROAIW Command 2–11
PROAPW Command 2–12
PROBIW Command 2–13
PROBRKR Command 2–14
PROQUIET Command 2–16
PROSERVE Command 2–17
PROSHUT Command 2–19
PROWDOG Command 2–23
PROWIN32 Command 2–24

3. Startup Parameter Usage Categories 3–1
3.1 About the Startup Parameters and Usage Categories 3–2
3.2 Client Session Parameters (CS) 3–2
3.2.1 Client Performance Parameters 3–3
3.2.2 Client Product-type Parameters 3–5
3.2.3 Progress 4GL Parameters 3–6
3.2.4 Client Internationalization Parameters 3–8
3.2.5 Session Statistics Parameters 3–10
3.2.6 Windows Session Parameters 3–11
3.3 Client Connection Parameters (CC) 3–12
3.3.1 Client Database Connection Parameters 3–12
3.4 Progress Database Server Parameters (DBS) 3–15
3.4.1 Database Server Performance Parameters 3–16
3.4.2 Database Server-type Parameters 3–18
3.4.3 Database Server Internationalization Parameters 3–18
3.4.4 Database Server Statistics Collection Parameters 3–19
3.4.5 Database Server Network Parameters 3–20
3.5 DataServer Parameters (DS) 3–22
3.6 Open Interface Driver Parameters (OID) 3–23

4. Startup Parameter Descriptions 4–1
Single-user Mode (-1) 4–2
AdminServer Port (-adminport) 4–3
After-image Buffers (-aibufs) 4–4
After-image Stall (-aistall) 4–5
Async Queue Size (-asyncqueuesize) 4–6
Blocks in Database Buffers (-B) 4–7
Batch (-b) 4–8
ADE r-code Location (-baseADE) 4–9
Base Index (-baseindex) 4–11
Registry Base Key (-basekey) 4–12
Base Table (-basetable) 4–13
Before-image Buffers (-bibufs) 4–14
Threshold Stall (-bistall) 4–15
Recovery Log Threshold (-bithold) 4–16
Private Read-only Buffers (-Bp) 4–17
Maximum Private Buffers per Connection (-Bpmax) 4–18
Bleeding Record Lock (-brl) 4–19
Buffer Size for Temporary Tables (-Bt) 4–20
Index Cursors (-c) 4–21
Schema Cache File (-cache) 4–22
Check Double-byte Enabled (-checkdbe) 4–24
SQL-92 Server Java Classpath (-classpath) 4–25
Client Logging (-clientlog) 4–26
Conversion Map (-convmap) 4–28
Communications Parameter File (-cp) 4–29
Case Table (-cpcase) 4–30
Collation Table (-cpcoll) 4–31
Internal Code Page (-cpinternal) 4–32
Log File Code Page (-cplog) 4–33
Print Code Page (-cpprint) 4–34
R-code in Code Page (-cprcodein) 4–35
R-code out Code Page (-cprcodeout) 4–36
Stream Code Page (-cpstream) 4–37
Terminal Code Page (-cpterm) 4–38
Cursor Size (-cs) 4–39
Directory Size (-D) 4–40
Date Format (-d) 4–41
DataService (-DataService) 4–42
Physical Database Name (-db) 4–43
Debugger (-debug) 4–44
Debug Alert (-debugalert) 4–45
Dictionary Expressions (-dictexps) 4–46
Direct I/O (-directio) 4–47
DataServer (-Dsrv) 4–48

Database Type (-dt) 4–49
European Numeric Format (-E) 4–50
ESQL Log (-esqllog) 4–51
ESQL No Padding (-esqlnopad) 4–52
Event Level (-evtlevel) 4–53
Expand Browse (-expandbrow) 4–54
Force Access (-F) 4–55
Schema Field Cache Size (-fc) 4–56
Field List Disable (-fldisable) 4–57
Filtering Asynchronous COM Events (-filterocxevents) 4–58
Before-image Cluster Age (-G) 4–59
Group Delay (-groupdelay) 4–60
Host Name (-H) 4–61
Number of Databases (-h) 4–62
Hardlimit (-hardlimit) 4–63
Hash Table Entries (-hash) 4–64
No Crash Protection (-i) 4–65
Dynamics Parameter (-icfparam) 4–67
Index Range Size (-indexrangesize) 4–68
Initialization File (-ininame) 4–69
Input Characters (-inp) 4–70
AS/400 Ignore Stamp (-is) 4–71
Initial Value Segment No Convert (-isnoconv) 4–72
Keyword Forget List (-k) 4–73
Lock Table Entries (-L) 4–74
Local Buffer Size (-l) 4–75
Logical Database Name (-ld) 4–76
Lock Timeout (-lkwtmo) 4–77
Language (-lng) 4–78
Auto Server (-m1) 4–79
Manual Server (-m2) 4–80
Secondary Login Broker (-m3) 4–81
Maximum Clients Per Server (-Ma) 4–82
Maximum Dynamic Server (-maxport) 4–83
Delayed BI File Write (-Mf) 4–84
Minimum Clients per Server (-Mi) 4–85
Minimum Dynamic Server (-minport) 4–86
Message Buffer Size (-Mm) 4–87
Maximum Memory (-mmax) 4–88
Maximum Servers (-Mn) 4–89
Servers Per Protocol (-Mp) 4–90
Maximum Servers Per Broker (-Mpb) 4–91
VLM Page Table Entry Optimization (-Mpte) 4–92
Record Buffer Size (-Mr) 4–93
Shared-memory Overflow Size (-Mxs) 4–94

Network Type (-N) 4–95
Number of Users (-n) 4–96
Nested Blocks (-nb) 4–97
No Lock (-NL) 4–98
No Check Temp-table Names (-nochkttnames) 4–99
Index Hint (-noindexhint) 4–100
Server Join (-nojoinbysqldb) 4–101
SELECT Pass Through Disable (-noSQLbyserver) 4–102
Fractional Separator (-numdec) 4–103
Thousands Separator (-numsep) 4–104
Printer (-o) 4–105
Outer-join Mode (-ojmode) 4–107
Password (-P) 4–108
Startup Procedure (-p) 4–109
Parameter (-param) 4–110
Parameter File (-pf) 4–111
Pin Shared Memory (-pinshm) 4–112
PROLIB Memory (-plm) 4–113
PROLIB Swap (-pls) 4–114
Fast Schema Change (-populate) 4–115
ESQL PROPATH (-pp) 4–116
Configuration Properties File (-properties) 4–117
Proxy Host (-proxyhost) 4–118
Proxy Password (-proxyPassword) 4–119
Proxy Port (-proxyport) 4–120
Proxy Userid (-proxyUserid) 4–121
ANSI SQL (-Q) 4–122
ANSI SQL Client (-Q2) 4–123
Quick Request (-q) 4–124
Non-reliable I/O (-r) 4–125
Alternate Random Number Generator (-rand) 4–126
Reread Nolock (-rereadnolock) 4–127
Run 4GL Client (-rg) 4–129
Read-only Media (-RO) 4–130
License Usage Reporting Interval (-rptint) 4–131
Run Query Client (-rq) 4–133
Run Run-time Client (-rr) 4–134
Encrypted Compiler Mode (-rx) 4–135
Service Name (-S) 4–136
Stack Size (-s) 4–138
Semaphore Sets (-semsets) 4–139
Server Group (-servergroup) 4–140
Type of Server to Start (-ServerType) 4–141
AS/400 Server Program Name (-Sn) 4–142
Spin Lock Retries (-spin) 4–143

SQL-92 Stack Size (-SQLStackSize) 4–145
Statement Cache Size (-SQLStmtCacheSize) 4–146
Stash Area (-stsh) 4–147
Open Interface Driver Connection (-SV) 4–148
Temporary Directory (-T) 4–150
Save Temp Files (-t) 4–152
Table Range Size (-tablerangesize) 4–153
Speed Sort (-TB) 4–154
Merge Number (-TM) 4–155
Token (-tok) 4–156
Trigger Location (-trig) 4–157
User ID (-U) 4–158
Use OS Locale (-useOsLocale) 4–159
User Count (-usrcount) 4–160
Version 6 Colon (-v6colon) 4–162
Version 6 Query (-v6q) 4–163
Windows Passthrough Printing (-Wa -wpp) 4–164
Windows Single Session (-wss) 4–165
Windows Exit - No Dialog (-wy) 4–166
Statistics (-y) 4–167
Statistics with CTRL-C (-yc) 4–169
Segment Statistics (-yd) 4–170
Four Digit Year Default (-yr4def) 4–171
Statistics With Cross-reference (-yx) 4–172
Century Year Offset (-yy) 4–175

Figures

Figure 4–1: Sample Output From -yx Startup Parameter 4–173

Tables

Table 1–1: Control Characters In Parameter Files 1–7
Table 2–1: Progress Command Components 2–2
Table 2–2: UNIX and Windows Startup and Shutdown Commands 2–3
Table 2–3: Windows-only Startup Commands 2–5
Table 3–1: Startup Parameter Usage Categories 3–2
Table 3–2: Client Performance Parameters 3–3
Table 3–3: Client Product-type Parameters 3–5
Table 3–4: Client 4GL Parameters 3–6
Table 3–5: Client Internationalization Parameters 3–8
Table 3–6: Session Statistics Parameters 3–10
Table 3–7: Windows Session Parameters 3–11
Table 3–8: Client Database Connection Parameters 3–12
Table 3–9: Server Performance Parameters 3–16
Table 3–10: Server-type Parameters 3–18
Table 3–11: Server Internationalization Parameters 3–18
Table 3–12: Server Statistics Collection Parameters 3–19
Table 3–13: Server Network Parameters 3–20
Table 3–14: DataServer Parameters 3–22
Table 3–15: ESQL-89 Client Parameters 3–23
Table 4–1: Parameter Interpretation With Service Name (-S) 4–137
Table 4–2: Parameters Used With the -SV Parameter Progress 4–149
Table 4–3: Attributes Set By the Use OS Locale (–useOsLocale) Parameter 4–159
Table 4–4: Century Setting and Date Format 4–175

Preface

Purpose

This book provides business application developers and system administrators for Progress with a comprehensive reference of the startup and shutdown commands and the startup parameters.

Audience

The primary audiences for this book are business application developers and Progress system administrators. The secondary audiences are Progress developers and technical support engineers.

Organization Of This Manual

Chapter 1, “Introduction”

Introduces the startup commands and parameters.

Chapter 2, “Startup and Shutdown Commands”

Describes the Progress commands for starting up and shutting down Progress sessions and processes.

Chapter 3, “Startup Parameter Usage Categories”

Provides an overview of the Progress startup parameters organized by usage category.

Chapter 4, “Startup Parameter Descriptions”

Describes the Progress startup parameters in detail, in alphabetical order by syntax.

How To Use This Manual

Use the introductory material in chapters 1, 2, and 3 to become familiar in general with the commands and parameters. Then use the quick reference tables in chapters 2 and 3 to find the command or parameter that you need. If you need additional information about a parameter, refer to the descriptions in chapter 4.

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 path names

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)
```

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

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

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

1 ♦ Start the Progress Procedure Editor:

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

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

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

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

Other Useful Documentation

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

Getting Started

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

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

Progress Installation and Configuration Guide Version 9 for UNIX

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

Progress Installation and Configuration Guide Version 9 for Windows

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

Progress Version 9 Product Update Bulletin

A 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.

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 for 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.

1

Introduction

This chapter introduces the commands and parameters that you can use to start up and shut down Progress processes. It also describes how to use a parameter file (.pf) to maintain startup parameters for a particular database startup, client session startup, or database connection. The following topics are included:

- About the Startup Commands
- About the Startup Parameters
- Using Parameter Files

1.1 About the Startup Commands

Progress provides startup commands that support its client/server architecture and a wide range of configuration options. In client/server configurations, the client application and the database server are separate processes. The Progress user interface and Progress applications execute in a client session, sending requests to the Progress server. The Progress server accesses the database on behalf of each client session.

Consequently, the startup commands fall into two general categories:

- Commands to startup and shutdown database processes, such as database servers, database brokers, and background writers
- Commands to start up and shutdown client sessions that connect to databases, such as a 4GL client, AppServer, and DataServer client

After setting up your database, you are ready to run Progress. In general, you start clients and servers as follows:

- Start a database server or broker for a database. Starting a server or broker for a database is distinct from connecting to a database. For more information on starting these processes, see the *Progress Installation and Configuration Guide Version 9 for UNIX* or *Progress Installation and Configuration Guide Version 9 for Windows*.
- Start the client applications.

For more information about the startup commands, see Chapter 2, “Startup and Shutdown Commands.”

1.2 About the Startup Parameters

You use startup parameters with the startup commands to help define the context in which the database processes and client sessions run. The requirements and use of parameters varies with the different operating systems and network environments. For example, you can use the startup parameters to control how a client connects to a database and enhance database server and client performance. These parameters are organized into the following usage categories:

- Client Session
- Client Connection
- Progress Database Server
- DataServer
- Open Interface Driver

For information about the startup parameter usage categories, see "Chapter 3, "Startup Parameter Usage Categories." For details about each parameter, see Chapter 4, "Startup Parameter Descriptions."

1.3 Using Parameter Files

A *parameter file* is an operating system file that contains one or more startup parameters. The primary reason to use a parameter file is to avoid typing commands each time you want to start Progress. Instead of repeatedly typing a series of parameters and other arguments, type them once and place them in a parameter file. You can also use a parameter file to maintain the startup parameters for a particular database, group of users, or system configuration. A parameter file has a .pf extension.

Every Progress executable invokes a parameter file named `startup.pf` by default. This file is located in the DLC directory. It must exist for Progress to execute properly. You can modify `startup.pf` to add parameter definitions. You can also create one or more additional parameter files and direct Progress to invoke them after it invokes `startup.pf`.

NOTE: In Windows, you can specify the name of the Progress startup file using the PROSTARTUP environment variable in the `progress.ini` file and the Registry. See the *Progress Client Deployment Guide* for more information on the `progress.ini` file.

1.3.1 Calling Parameter Files

To call an additional parameter file, use the Parameter File (-pf) startup parameter in the Progress startup command line or in a CONNECT statement.

Calling a Parameter File In a Startup Command

From a startup command line, use the Parameter File (-pf) startup parameter to call the parameter file. Enter one of the following commands:

Operating System	Syntax
UNIX	`pro sports -pf filename.pf`
Windows[1]	`prowin32 -1 sports -pf filename.pf`

[1] Specify in the Command Line field of a Shortcut Properties dialog box.

The *filename* can be any valid filename. Progress combines all parameters on a command line with those in the parameter file. If the same parameter is mentioned more than once, the last occurrence takes precedence. For more information on startup parameters, see Chapter 3, "Startup Parameter Usage Categories"and Chapter 4, "Startup Parameter Descriptions."

Calling a Parameter File In a CONNECT Statement

Use the following syntax to specify a parameter file in a CONNECT statement in a Progress 4GL procedure:

```
CONNECT -pf sports.pf
```

For example, this procedure connects using a parameter file called `new_york.pf`:

new_york.p

```
UPDATE ny_user ny_password BLANK.
  CONNECT -pf new_york.pf
    -U VALUE(ny_user) -P VALUE(ny_passwd)
NO-ERROR.
  IF NOT CONNECTED("new_york") THEN
    MESSAGE "Connection to New York office failed".
```

Calling Multiple Parameter Files

You can use the -pf parameter as many times as you like on a command line. This lets you specify application-specific parameters in one parameter file, database-specific parameters in a second parameter file, and user-specific parameters in yet another parameter file. It also allows you to connect to multiple databases from the same command line or CONNECT statement.

When connecting to more than one database, the startup parameters must appear after the corresponding Physical Database Name (-db) parameter and database name. If connecting to more than one database, consider putting all the database-related parameters in a parameter file, then add the -db parameter to the top of the file. To support earlier releases, Progress assigns any startup parameters that precede all database names to the first database on the command line.

Progress processes parameters from left to right. Each -pf parameter encountered is processed first before the next top-level parameter is evaluated. If a parameter file specified by -pf contains another -pf parameter, the nested parameter file is evaluated before resuming the top-level processing.

NOTE: If the same parameter is mentioned more than once, the last occurrence takes precedence.

1.3.2 Creating and Editing Parameter Files

You can use any text editor to create a file containing startup parameters. Follow the format explained in the "Parameter File Format" section.

You can also use Progress tools to create and edit parameter files. Follow these steps:

1 ♦ Access the Data Administration tool or Data Dictionary.

2 ♦ Choose Utilities→ Editor for Parameter Files. Progress prompts you for the file you want to create or edit.

3 ♦ Specify the filename. Progress alphabetically lists all available startup parameters.

4 ♦ Specify which of the parameters you want in the file by choosing the parameter and entering yes, no, or the appropriate value.

5 ♦ Enter any comments you want to include in the parameter file. After creating the parameter file, Progress returns you to the Data Administration or Data Dictionary main window.

1.3.3 Parameter File Format

The following is a sample parameter file:

new_york.pf

```
-db  /usr/disk0/newyork          # new york central db

-B   200                         # database buffers
-H   nysystem -S nyserver        # host, service
-L   10000                       # lock table entries

#    All other parameters receive their default values.
```

The format of the parameter file is the same for all operating systems. Follow these rules when creating a parameter file:

- Place a parameter and its argument on the same line.
- Use a maximum of 40 tokens per line in a parameter file.
- Specify parameters with the UNIX syntax.
- Use a pound sign (#) to begin a comment. Progress ignores the rest of the line.
- Enclose a value in single (’) or double (“) quotes. Any white space inside the quotes is treated as part of the token.
- Any parameter not named in a parameter file receives a default value. To remind yourself of the default values, consider specifying all parameters and their values in the parameter file and adding the comment “# default” to those values that do not change.
- Use a tilde (~) to suppress the special meaning of the single (’) or double (“) quotes or tilde (~). A tilde also prefixes the control characters listed in Table 1–1 that Progress can recognize in a parameter file.

Table 1–1: Control Characters In Parameter Files

Character	Description	Octal Code
~*nnn*	The character having octal code *nnn*	–
~b	BS character	~010
~t	HT (horizontal tab) character	~011
~n	LF (line feed) character	~012
~r	VT (carriage return) character	~013
~f	FF (form feed) character	~014
~E	ESC character	~033

On UNIX, use the backslash (\) in place of a tilde.

2

Startup and Shutdown Commands

This chapter presents the Progress startup and shutdown commands that you can enter at the command line. It describes the syntax of a startup command, summarizes the tasks you can perform with startup commands, and then describes each command and its parameters. For more information on how parameters are used, see Chapter 3, “Startup Parameter Usage Categories.” For details on each parameter, see Chapter 4, “Startup Parameter Descriptions.”

2.1 Startup Command Syntax

These are the conventions used in command syntax descriptions:

```
command [ db-name ] [ parameter ] [ value ] ...
```

For example, you can use the following command to allow 100 users to access the sports database and then set values for the database connection, performance, and network parameters.

```
proserve sports -n 100 -B 30000 -L 1000 -S sprtsv -H sys27
```

Table 2–1 describes each of the command components.

Table 2–1: Progress Command Components

Component	Description
command	On UNIX, the command runs a script that executes a Progress executable with appropriate parameters. On Windows, some commands run a batch file that executes a Progress executable with appropriate parameters. Other commands run a Progress executable directly.
db-name	Name of the database you want to connect to.
parameter	Operating criteria for the command.
value	Numeric value or file specification for the parameter.

Observe the following conventions when you enter a command:

- On UNIX, use lowercase characters for commands; on Windows, commands are not case sensitive.
- On UNIX and Windows, enter parameters exactly as shown in the syntax descriptions.
- On UNIX, values can be case sensitive. (For example, names of UNIX files are case sensitive.) In addition, on Windows NT using the NT File system (NTFS), values can be case sensitive. On Windows, values are not case sensitive using the File Allocation Table (FAT or FAT32).

2.2 Tasks You Can Perform With Startup and Shutdown Commands

Startup commands start up and shut down Progress database or client processes. Table 2–2 summarizes the tasks you can perform on UNIX and Windows systems and the related startup and shutdown commands.

Table 2–2: UNIX and Windows Startup and Shutdown Commands *(1 of 2)*

Task	Command
Start a character Progress Procedure Editor and connect to a single-user database	`pro` *db-name*
Start a character Progress Procedure Editor and connect to a multi-user database	`mpro` *db-name* `-S` *service-name* `-H` *host-name* `-N` *network-type*
Start a Progress character client session in batch mode and connect to a single-user database	`bpro` *db-name* `-p` *procedure-name*
Start a Progress character client session in batch mode and connect to a multi-user database	`mbpro` *db-name* `-p` *procedure-name* `-S` *service-name* `-H` *host-name* `-N` *network-type*
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*

Table 2–2: UNIX and Windows Startup and Shutdown Commands *(2 of 2)*

Task	Command
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*
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] Choose option 1 (Disconnect a User) to disconnect the process.	`proshut` *db-name*

[1] Option available only on Enterprise product

Table 2–3 summarizes tasks and their related startup commands for Windows only. You can use these commands in the Command Line field of a Shortcuts Properties dialog box.

Table 2–3: Windows-only Startup Commands

Task	Command
Start a GUI Progress Procedure Editor and connect to a single-user database	prowin32 -1 *db-name*
Start a GUI Progress Procedure Editor and connect to a multi-user database	prowin32 *db-name* -S *service-name* -H *host-name* -N *network-type*
Start the ADE Desktop and connect to a single-user database	prowin32 -p _desk.p -1 *db-name*
Start the ADE Desktop and connect to a multi-user database	prowin32 -p _desk.p -1 *db-name* -S *service-name* -H *host-name* -N *network-type*
Start a Progress batch session and connect to a single-user database	prowin32 -b -1 *db-name* -p *procedure*
Start a Progress batch session and connect to a multi-user database	prowin32 -b *db-name* -p *procedure* -S *service-name* -H *host-name* -N *network-type*

2.3 Startup and Shutdown Command Descriptions

The following sections describe the Progress startup commands in detail, in alphabetical order. Each section includes a table that shows the syntax of the command and other information about it. Following the table is a description of each parameter.

NOTE: In general, if the database is not in your current directory, you must specify the full pathname of the database. There are a few exceptions. See the description of the command you want to use for more information.

BPRO Command

Starts a single-user Progress client session in batch or background mode.

SYNTAX

Operating System	Syntax
UNIX Windows	bpro [*db-name*] -p *procedure-name* [*parameters*] [> *output-file*]

PARAMETERS

db-name

Specifies the database you want to connect.

-p *procedure-name*

Specifies a procedure to run at startup.

parameters

Specifies the Progress startup parameters you want to use.

output-file

Specifies the name of the file where messages are sent.

NOTES

- For UNIX, the BPRO command runs the following executable:

```
_progres -1 -b &
```

  The & character causes a Progress batch session to run in the background, and returns control to the terminal once the session starts. When the & character is not present, for example, when you simply use the (-b) startup parameter, Progress initiates a batch session in the foreground without terminal interaction and control does not return to the terminal until the session completes.

- On UNIX and Windows NT, you can redirect batch job input and output with the < and > redirection symbols. You can also use the pipe symbol (|) to put a Progress batch run in a command pipeline.

MBPRO Command

Starts a multi-user Progress client session in batch or background mode.

SYNTAX

Operating System	Syntax
UNIX Windows	mbpro [*db-name*] -p *procedure-name* [*parameters*] [> *output-file*]

PARAMETERS

db-name

Specifies the database you want to connect. If you are connecting to a database through shared memory and the database is not in your current directory, you must specify the path of the database. On the other hand, if you are making a client/server connection to the database, you do not need to specify a path.

-p *procedure-name*

Specifies the procedure to run at startup.

parameters

Specifies the startup parameters you want to use.

output-file

Specifies a file where output messages are sent.

NOTES

- On UNIX, the MBPRO command runs the following executable:

```
_progres -b &
```

- On UNIX and Windows NT, you can redirect batch job input and output with the < and > redirection symbols. You can also use the pipe symbol (|) to put a Progress batch run in a command pipeline.

MPRO Command

Starts a multi-user mode character Progress client session.

SYNTAX

Operating System	Syntax
UNIX Windows	`mpro [ db-name ] [ parameters ]`

PARAMETERS

db-name

Specifies the database you want to connect to. If you are connecting to a database through shared memory and the database is not in your current directory, you must specify the path of the database. On the other hand, if you are making a client/server connection to the database, you do not need to specify a path.

parameters

Specifies the startup parameters you want to use.

NOTES

- The MPRO command runs the following executable:

```
_progres
```

- The server must be running before you issue this command with the *db-name* parameter. If the server is not running, Progress displays the following error message:

```
There is no server for database db-name (1423)
```

PRO Command

Starts a single-user mode character Progress client session.

SYNTAX

Operating System	Syntax
UNIX Windows	`pro [ db-name ] [ parameters ]`

PARAMETERS

db-name

Specifies the database you want to start.

parameters

Specifies the parameters you want to use. You can use any of the single-user startup parameters with the PRO command.

NOTE

- The PRO command runs the following executable:

```
_progres -1
```

PROAIW Command

Starts the after-image writer (AIW) process. The AIW improves performance on shared-memory systems by performing overhead after-image operations in the background. For more information on the AIW, see the *Progress Database Administration Guide and Reference*.

SYNTAX

Operating System	Syntax
UNIX Windows	`proaiw db-name`

PARAMETERS

db-name

Specifies the database where you want to start the AIW process.

NOTES

- To stop the AIW, disconnect it by using thePROSHUT Command. You can start and stop the AIW at any time without shutting down the database.
- The AIW counts as one user. You might have to increment the value of the Number of Users (-n) parameter to allow for the AIW. However, the AIW does not count as a licensed user.
- You can increase the number of buffers in the after-image buffer pool by using the After-image Buffers (-aibufs) parameter. Increasing the number of buffers when running an AIW increases the availability of empty buffers to client and server processes. Increasing the After-image Buffers parameter has no effect if the AIW is not running.

PROAPW Command

Starts an asynchronous page writer (APW). APWs improve database performance on shared-memory systems by performing overhead operations in the background. For more information on APWs, see the *Progress Database Administration Guide and Reference*.

SYNTAX

Operating System	Syntax
UNIX Windows	`proapw` *db-name*

PARAMETERS

db-name

Specifies the database where you want to start an APW.

NOTES

- To stop an APW, disconnect it by using the PROSHUT Command. You can start and stop APWs at any time without shutting down the database.
- Each APW counts as a user. You might have to increase the value of the Number of Users (-n) parameter to allow for APWs. However, APWs do not count as licensed users.
- The optimal number depends on your application and environment. To start, use one page writer for each disk where the database resides. If this is insufficient, add more. For an application that performs many updates, start one APW for each disk containing your database, plus one additional APW. Applications that perform fewer changes to a database require fewer APWs.

PROBIW Command

Starts a before-image writer (BIW) process. The BIW improves database performance on shared-memory systems by performing before-image overhead operations in the background. For more information on the BIW, see the *Progress Database Administration Guide and Reference*.

SYNTAX

Operating System	Syntax
UNIX Windows	`probiw` *db-name*

PARAMETERS

db-name

Specifies the database where you want to start a BIW.

NOTES

- To stop the BIW process, disconnect it by using the PROSHUT Command. You can start and stop the BIW at any time without shutting down the database.
- The BIW process counts as one user. You might have to increment the value of the Number of Users (-n) parameter to allow for the BIW. However, the BIW does not count as a licensed user.
- You can increase the number of before-image buffers with the Before-image Buffers (-bibufs) parameter. Increasing the number of buffers increases the availability of empty buffers to client and server processes.

PROBRKR Command

Starts the DataServer broker, which is used when Progress clients connect to non-Progress databases. To use the DataServer from a remote client, you must first start the broker. Once you start the broker, it can receive the client requests and spawn the appropriate DataServer.

SYNTAX

Operating System	Syntax
UNIX Windows	probrkr [*db-name*] -S *service-name* [-H *host-name*] [-N *network-type*]

PARAMETERS

db-name

Specifies the name of the database you are connecting to.

service-name

Specifies the name of the broker process on the host machine.

host-name

Specifies the name of the machine where the DataServer broker is installed. This parameter is only required on UNIX System V systems using the TLI network type. The default value is the current host.

network-type

Specifies the network protocol for which the broker will run.

Progress supports TCP, SNA (Progress/400 only), and LFP (Progress/400 only).

NOTES

- You can use any of the startup parameters with the PROBRKR Command.
- For information about connecting to a non-Progress database, remotely or locally, see the DataServer documentation.
- You must start the remote broker in the same environment where the non-Progress database is, because the servers it spawns inherit the setup of the environment from the broker. For example, if you use an ORACLE database on a UNIX platform, set the ORACLE_SID value for that environment, as you would for a local DataServer.
- Start the broker on a node that is locally connected to the disk containing the non-Progress database.

PROQUIET Command

Stops all writes to database files by enabling a “quiet” processing point (useful for advanced backup strategies). You can also use the PROQUIET command with the bithreshold parameter to adjust the size of the recovery log threshold online. Use the -bithold startup parameter to set the size of the recovery log threshold on startup.

For more information on using database quiet points, see the *Progress Database Administration Guide and Reference*.

SYNTAX

Operating System	Syntax
UNIX Windows	`proquiet` *db-name* `-C { { enable | disable } | bithreshold` *n* `}`

PARAMETERS

db-name

Specifies the name of the database where you are enabling or disabling a quiet processing point.

`-C`

Specifies a parameter.

`enable | disable`

Enables or disables a quiet processing point. Any processes that attempt transactions while a quiet point is enabled must wait until the quiet point is disabled.

`bithreshold` *n*

Specifies the maximum size to which .bi recovery files can grow, where *n* is an integer specifying the size of the threshold in MB. You can increase the size of the threshold above the current value or reduce the size to one cluster larger than the size of the recovery log file at the time the PROQUIET command is issued. You must issue the PROQUIET enable command to enable a quiet point before using the -bithreshold parameter to adjust the size of the threshold.

PROSERVE Command

Starts the initial Progress multi-user server process. This server process coordinates all access to the specified Progress database.

SYNTAX

Operating System	Syntax
UNIX Windows	`proserve { db-name | -servergroup servergroup-name } [ parameters ]`

PARAMETERS

db-name

Specifies the specific database you want to start Progress against.

`-servergroup` *servergroup-name*

Specifies the logical collection of server processes to start. The *servergroup-name* you specify must match the name of a servergroup in the `conmgr.properties` file. You create servergroups using the Progress Explorer tool, which saves them in the `conmgr.properties` file. For more information, see the *Progress Installation and Configuration Guide Version 9 for Windows* or *Progress Installation and Configuration Guide Version 9 for UNIX*.

parameters

Specifies the startup parameters for the broker/server. If both -servergroup and parameters are specified, those that occur last on the command line take precedence.

NOTES

- The PROSERVE command runs the following executable:

```
_mprosrv
```

- The initial database server process is called the *broker*. The process manages shared resources and starts servers for remote users, if necessary.

- You can specify only one database name when using PROSERVE to start a server or servergroup.
- Servergroups manage network connections four separate ways:
  - Accept no network connections
  - Accept SQL-92 and 4GL network connections
  - Accept only SQL-92 network connections
  - Accept only 4GL network connections

  Typically, servergroups share common attributes such as connection port, number of servers, and how connected clients are distributed among the severs.
- You create servergroups using the Progress Explorer tool, which saves them in the `conmgr.properties` file. The *servergroup-name* you specify with the PROSERVE -servergroup parameter must match the name of a servergroup in the `conmgr.properties` file. Do not edit the `conmgr.properties` file directly. Instead, use Progress Explorer.
- The behavior of the -servergroup parameter is similar to the behavior of the -pf (parameter file) parameter. In effect, -servergroup causes a server to load the parameters associated with the servergroup, including the database name.

  It is possible to override the parameter values associated with a servergroup by adding additional parameters to the PROSERVE command. For example, if the database buffer pool is set to 10,000 within the configuration associated with a servergroup, you can specify a larger value by adding an additional parameter:

```
proserve -servergroup sports2000.myconfig.4GLdefault -B 20000
```

  Conversely, if you specify a startup parameter before the -servergroup parameter, the startup parameter can be overridden when the same parameter is set in the servergroup configuration file. For example, if you place the additional parameter before the -servergroup parameter, the database buffer pool remains 10,000:

```
proserve -B 20000 -servergroup sports2000.myconfig.4GLdefault
```

PROSHUT Command

Shuts down the Progress database server and individual Progress processes. Before you shut down the server, have all application users quit their Progress sessions. If necessary, you can disconnect users by using the PROSHUT command's Disconnect a User or Unconditional Shutdown parameters.

SYNTAX

Operating System	Syntax
UNIX Windows	proshut *db-name* [-b | -by | -bn | -C list | -C disconnect *usernum* | -cpinternal *codepage* | -cpstream *codepage* | -F | -Gw | -H *host-name* | -N *network-type* | -S *service-name*] ...

PARAMETERS

db-name

Specifies the database the server is running against.

-b

Directs Progress to perform a batch shutdown. When no client is connected, the database automatically shuts down. When one or more clients are connected, PROSHUT prompts the user to enter “yes” to perform an unconditional batch shutdown and to disconnect all active users; or “no” to perform a batch shutdown only if there are no active users. The -b parameter combines the functionality of the -by or -bn parameters.

-by

Directs Progress to perform an unconditional batch shutdown and to disconnect all active users. Using -by in conjunction with -F causes an emergency shutdown.

-bn

Directs Progress to perform a batch shutdown only if there are no active users.

-C list

Lists all of the users connected to the database. The list is printed out to the screen without any page breaks.

-C disconnect *usernum*

Allows you to initiate a disconnect for the specified user. This is similar to option 1 of the PROSHUT menu.

-cpinternal *codepage*

Identifies the code page that Progress uses in memory.

-cpstream *codepage*

Identifies the code page used for stream I/O.

-F

Starts an emergency shutdown. To use this parameter, you must run PROSHUT on the machine where the server resides. This parameter is not applicable for remote shutdowns or DataServer shutdowns.

-Gw

For DataServers, specifies the DataServer broker to shut down.

-H *host-name*

Specifies the machine where the database server runs. You must specify the host name if you issue the shutdown command from a machine other than the host.

-N *network-type*

Specifies the networking protocol used to connect to the database server. You must specify the network type if you issue the shutdown command from a machine other than the host.

-S *service-name*

Specifies the database server or broker process. You must specify the service name if you issue the shutdown command from a machine other than the host.

When you enter the PROSHUT command without the -by, -bn, or -F parameters, the following menu appears:

```
1 Disconnect a User
2 Unconditional Shutdown
3 Emergency Shutdown (Kill All)
x Exit
```

The following table lists the menu options and their actions:

Option	Action
1	Prompts you for the number of the user you want to disconnect.
2	Disconnects all users and shuts down the database.
3	Prompts you to confirm your choice. If you cancel the choice, you cancel the shutdown. If you confirm the choice, Progress displays the following message: `Emergency shutdown initiated...` PROSHUT marks the database for abnormal shutdown, kills all remaining processes connected to the database, and deletes shared-memory segments and semaphores. The database is in a crashed state. Progress performs normal crash recovery when you restart the database and backs out any active transactions.
x	Cancels the shutdown without taking any action.

NOTES

- The PROSHUT command runs the following executable:

```
_mprshut
```

- You can access PROSHUT from the PROMON utility by using the Shut Down Database qualifier.
- The user who shuts down the server must have started it, or be root (on UNIX).
- When you initiate PROSHUT over a network, the amount of time that it takes to actually shut down all of the Progress processes and to free any ports varies depending on the number of clients, brokers, and servers that must be shut down. The PROSHUT utility might return control to the terminal before all of the processes are stopped and resources are freed.
- If you specified a new value for -cpinternal or -cpstream when you opened the database, you must specify that same value for -cpinternal or -cpstream when you close the database with the PROSHUT command. If you do not, PROSHUT uses the values for -cpinternal and -cpstream found in the main startup parameter file created during installation (such as `DLC/startup.pf`). If the values of -cpinternal or -cpstream specified for your database do not match the values specified in the main startup parameter file, you receive the following error message:

```
Code page conversion table for table-name to table-name was not found.
(6063)
```

PROWDOG Command

Starts the Progress Watchdog process. The Watchdog cleans up after improperly terminated Progress processes. For example, it releases any locks or shared-memory structures that those processes might hold.

SYNTAX

Operating System	Syntax
UNIX Windows	`prowdog` *db-name*

PARAMETERS

db-name

Specifies the database to clean up after.

NOTES

- If the Watchdog finds a process that is no longer active, it releases all the appropriate record locks, backs out any live transactions, releases any shared-memory locks, and closes the connection. If the lost process is a server, it disconnects and cleans up all appropriate remote clients.
- If the process was changing shared memory when it terminated, shared memory is in an inconsistent state; the Watchdog forces a shutdown to protect the database.
- The Watchdog cannot detect lost remote clients because remote clients are not associated with a process. Instead, a network protocol timeout mechanism notifies the server that the network connection was lost.

PROWIN32 Command

Starts a graphical Progress session.

SYNTAX

Operating System	Syntax
Windows	`prowin32 [ db-name ] [ parameters ]`

PARAMETERS

db-name

Specifies the database you are using.

parameters

Specifies the session or connection parameters you want to use.

NOTE

- See the *Progress Client Deployment Guide* for more information about starting Progress sessions and connecting to databases.

3

Startup Parameter Usage Categories

This chapter provides an overview of the Progress startup parameters organized by usage categories. The chapter includes the following topics:

- About the Startup Parameters and Usage Categories
- Client Session Parameters (CS)
- Client Connection Parameters (CC)
- Progress Database Server Parameters (DBS)
- DataServer Parameters (DS)
- Open Interface Driver Parameters (OID)

Use this chapter as a quick reference. For a complete description of each startup parameter listed in alphabetical order, see Chapter 4, “Startup Parameter Descriptions.”

3.1 About the Startup Parameters and Usage Categories

You can change the default startup parameters by using startup parameters on a command line or by incorporating them into a `.BAT` file or a UNIX script file. You can also use a Progress parameter file, which can include any number of startup parameters. See Chapter 1, "Introduction" for information on the parameter file.

In this chapter, the startup parameters are organized into usage categories. Some parameters might be listed in more than one category. The syntax of the parameters is the same for UNIX and Windows unless noted otherwise. Table 3–1 defines the startup parameter usage categories.

Table 3–1: Startup Parameter Usage Categories

Usage Type	Purpose
Client Session (CS)	Use when starting any client session.
Client Connection (CC)	Use when connecting to a specific database.
Progress Database Server (DBS)	Use when starting a server or broker.
DataServer (DS)	Use when starting non-Progress DataServers.
Open Interface Driver (OID)	Use when starting the Open Interface Driver.

3.2 Client Session Parameters (CS)

Use the Client Session parameters when starting a session for a 4GL client, an ESQL-89 client, or an AppServer (as client). The parameters are organized alphabetically by syntax in the following subcategories:

- Client Performance
- Client Product-type
- Progress 4GL
- Client Internationalization
- Session Statistics
- Windows Session

3.2.1 Client Performance Parameters

Use the parameters listed in Table 3–2 to control client performance.

Table 3–2: Client Performance Parameters *(1 of 2)*

Parameter	Syntax	Purpose
Async Queue Size	-asyncqueuesize *n*	Specify the total number of bytes allowed for asynchronous request buffers allocated for a client
Buffer Size for Temporary Tables	-Bt *n*	Specify the size of the buffer pool for blocks or records from the temporary table.
Directory Size	-D *n*	Change the number (soft limit) of compiled procedure directory entries.
Schema Field Cache Size	-fc *num-entries*	Change the number (soft limit) of entries in the schema field cache.
Hardlimit	-hardlimit	Force Progress to adhere to set limits.
Local Buffer Size	-l *n*	Specify the size (soft limit) of the local record buffer.
Message Buffer Size	-Mm *n*	Specify the message buffer size for network protocols.
Maximum Memory	-mmax *n*	Set the initial amount (soft limit) of memory allocated for r-code segments.
Record Buffer Size	-Mr *n*	Specify the standard record buffer size in bytes.
PROLIB Memory	-plm	Allocate a 512-byte cache for the Progress library directory.
PROLIB Swap	-pls	Store r-code files locally in temporary sort files.

Table 3–2: Client Performance Parameters *(2 of 2)*

Parameter	Syntax	Purpose
Stack Size	-s *n*	Specify the stack size.
SQL-92 Stack Size	-SQLStackSize *n*	Specify the size, in MB, of the SQL Stack.
Statement Cache Size	-SQLStmtCacheSize *num_entries*	Specify the number of statements allowed in the cache.
Stash Area	-stsh *n*	Specify the number of 1 KB blocks to allocate in the stash area.
Temporary Directory	-T *dirname*	Specify the directory for temporary files.
Save Temp Files[1]	-t	Make temporary file entries visible.
Speed Sort	-TB *blocksize*	Specify the block size in kilobytes to allocate when sorting records for reports or rebuilding indexes.
Merge Number	-TM *n*	Specify the number of blocks or streams simultaneously merged during a sort.

[1] UNIX only.

3.2.2 Client Product-type Parameters

Use the parameters listed in Table 3–3 to control the nature of a client session.

Table 3–3: Client Product-type Parameters

Parameter	Syntax	Purpose
Run 4GL Client	-rg	Limit access or functionality of the session to the 4GL client.
Run Query Client	-rq	Limit access or functionality of the session to the Query client.
Run Run-time Client	-rr	Limit access or functionality of the session to the run-time client.
Encrypted Compiler Mode	-rx	Enable a query or run-time Progress version to compile encrypted source code and access the Data Dictionary to manage schema (for example: for security management and to dump or load the .df file).

3.2.3 Progress 4GL Parameters

Use the parameters listed in Table 3–4 to modify Progress 4GL behavior.

Table 3–4: Client 4GL Parameters *(1 of 3)*

Parameter	Syntax	Purpose
Batch	-b	Initiate a batch process with no terminal interaction.
ADE r-code Location	-baseADE	Specify the ADE r-code directory to the client so that the client can add this directory and all of the procedure libraries contained in the directory to the PROPATH.
Client Logging	-clientlog	Allows an application to automatically write all error and warning messages to the specified log file.
Directory Size	-D *n*	Change the number (soft limit) of compiled procedure directory entries.
Debugger	-debug	Start a Progress session by running the Debugger in stand-alone mode.
Debug Alert	-debugalert	Access 4GL stack trace information during a session.
Dictionary Expressions	-dictexps	Direct Progress to use dictionary and help validation for all fields in all frames when compiling.
Expand Browse	-expandbrow	Sets the EXPANDABLE and FIT-LAST-COLUMN option to on for all browsers created in the current session.
Filtering Asynchronous COM Events	-filterocxevents	Control the conditions under which you want asynchronous COM events handled.
Field List Disable	-fldisable	Direct Progress to ignore field lists in the r-code and fetch complete records.
Dynamics Parameter	-icfparam	Specify a character string that can be accessed from 4GL procedures within the Dynamics framework.

Table 3–4: Client 4GL Parameters *(2 of 3)*

Parameter	Syntax	Purpose
Input Characters	-inp *n*	Expand the available buffer space for a single statement.
Keyword Forget List	-k *filename*	Disable the keywords listed in *filename*.
Lock Timeout	-lkwtmo *seconds*	Specify a different “lock wait timeout.”
Nested Blocks	-nb *n*	Set the maximum number (soft limit) of nested procedure blocks.
No Lock	-NL	Make all record retrieval statements default to NO-LOCK record access.
No Check Temp-table Names	-nochkttnames	Suppress the limited checking of TEMP-TABLE column names when passing TEMP-TABLE as parameters to procedures.
Printer	-o *printername*	Identify the printer to use when processing OUTPUT TO PRINTER statements.
Outer-join Mode	-ojmode	Specify the mode in which mixed inner and left outer joins, in queries of three or more joined tables, are processed.
Startup Procedure	-p *filename*	Specify a procedure that executes when Progress starts.
Parameter	-param *string*	Supply information in a character string or comma-separated list of files to open in Procedure Editor buffers.
Proxy Host	-proxyhost	Specify the name of the host or the IP address of the host at which the HTTP-based proxy server is located.
Proxy Password	-proxyPassword	Authenticate the AppServer client to the HTTP-based proxy server.
Proxy Port	-proxyport	Specify the port on which the HTTP-based proxy server is listening.
Proxy Userid	-proxyUserid	Authenticate an AppServer client to the HTTP-based proxy server.

Table 3–4: Client 4GL Parameters *(3 of 3)*

Parameter	Syntax	Purpose
Quick Request	-q	Direct Progress to search PROPATH directories only on the first use of a procedure.
Alternate Random Number Generator	-rand *n*	Specify a different random number generator. Use the value 1 for the old generator or the value 2 for a new one.
Reread Nolock	-rereadnolock	Direct Progress to re-read the record from the database, even if the record is already in a buffer.
Token	-tok *n*	Specify the maximum number of tokens allowed in a 4GL statement.
Version 6 Colon	-v6colon	Direct Progress to use colon alignment of unlabeled fields (as in Version 6).
Version 6 Query	-v6q	Direct the Progress server to use only a single index to resolve FOR EACH statements (as in Version 6).

3.2.4 Client Internationalization Parameters

Use the parameters listed in Table 3–5 to control the format in which data appears.

Table 3–5: Client Internationalization Parameters *(1 of 3)*

Parameter	Syntax	Purpose
Check double-byte Enabled	-checkdbe	Display compile-time warnings on 4GL statements and functions that may impact the handling of double-byte or extended-character data.
Conversion Map	-convmap *filename*	Identify the conversion map file.
Case Table	-cpcase *tablename*	Identify the case table that establishes case rules for the code page.

Table 3–5: Client Internationalization Parameters *(2 of 3)*

Parameter	Syntax	Purpose
Collation Table	-cpcoll *tablename*	Identify a collation table to use with the code page.
Internal Code Page	-cpinternal *codepage*	Identify the code page that Progress uses in memory.
Log File Code Page	-cplog *codepage*	Identify the code page used for writing messages to the log file.
Print Code Page	-cpprint *codepage*	Identify the code page used for printer output.
R-code in Code Page	-cprcodein *codepage*	Identify the code page used for reading r-code text segments.
R-code out Code Page	-cprcodeout *codepage*	Identify the code page used for writing r-code text segments.
Stream Code Page	-cpstream *codepage*	Identify the code page used for stream I/O.
Terminal Code Page	-cpterm *codepage*	Identify the code page for character terminals.
Date Format	-d *dateform*	Specify the format for displaying dates, using the three-character string of d, m, y in any order.
European Numeric Format	-E	Display in European Numeric Format.
Initial Value Segment No Convert	-isnoconv	Disable a code page conversion that was not provided in earlier versions of Progress.
Language	-lng *language-name*	Identify the language with a character string, in quotes.
Fractional Separator	-numdec *numeric-value*	Change the character that represents a number’s decimal point in formatted text.

Table 3–5: Client Internationalization Parameters *(3 of 3)*

Parameter	Syntax	Purpose
Thousands Separator	-numsep *numeric-value*	Change the character that represents a number's thousands separator in formatted text.
Four Digit Year Default	-yr4def	Output a four-digit year from EXPORT, MESSAGE and PUT UNFORMATTED statements that may use a two-digit year.
Century Year Offset	-yy *n*	Specify the start of the 100-year period used to define the two-digit DATE value.

3.2.5 Session Statistics Parameters

Use the parameters listed in Table 3–6 to collect statistics about client session activity and performance.

Table 3–6: Session Statistics Parameters

Parameter	Syntax	Purpose
Statistics	-y	Collect procedure access and usage statistics.
Statistics with CTRL-C	-yc	Substitute CTRL-C for executing the SHOW-STATS statement.
Segment Statistics	-yd	Write segment statistics to the client monitor file (`client.mon` by default).
Statistics with Cross-reference	-yx	Collect procedure call statistics and write them to an output file (`proc.mon` by default).

3.2.6 Windows Session Parameters

Use the parameters listed in Table 3–7 to control a Windows session.

Table 3–7: Windows Session Parameters

Parameter	Syntax	Purpose
Registry Base Key [1]	-basekey *name*	Identify the registry basekey to use for environment information.
Event Level	-evtlevel *value*	Specify the level of information written to the NT Event Log.
Initialization File [1]	-ininame *name*	Identify the initialization registry sub-key or the application's initialization (.INI) file.
Windows Passthrough Printing [1]	-Wa -wpp	Enable Windows Passthrough Printing mode.
Windows Single Session	-wss	Limit a Windows GUI user to running one client session at a time.
Windows Exit - No Dialog	-wy	Prevent prompt for confirmation when Windows shuts down while Progress is active.

[1] You can only use this parameter on the command line. It is not supported in a parameter (.pf) file.

3.3 Client Connection Parameters (CC)

Use the Client Connection parameters when connecting to a Progress Database Server, AdminServer, AppServer, or a non-Progress DataServer. The parameters are organized alphabetically by syntax. For more information on connecting to a database, see the *Progress Client Deployment Guide*.

3.3.1 Client Database Connection Parameters

Use the parameters listed in Table 3–8 to control how you connect to a specified database.

If you use the Physical Database Name (-db) parameter to specify more than one database when you start a session, specify the connection parameters for each database after the database name to which they apply and before the next -db parameter, as shown. Progress applies database connection parameters only to the previously specified database:

NOTE: Progress implicitly specifies the -db parameter for the first database.

```
mpro [ -db] db1-name [ db1-parameters ] -db db2-name [ db2-parameters ]
```

You can specify all other parameters anywhere on the command line. If you specify the same parameter more than once, Progress uses the value you provide for the last instance of the parameter.

Table 3–8: Client Database Connection Parameters *(1 of 4)*

Description	Syntax	Purpose
Single-user Mode [1]	-1	Start Progress in single-user mode.
Blocks in Database Buffers [1]	-B *n*	Specify the number of database buffer blocks.
Private Read-only Buffers	-Bp *n*	Request a number of private read-only buffers.
Bleeding Record Lock	-brl	Hold a share lock on a released record used by more than one buffer.
Schema Cache File	-cache *filename*	Read the database schema from a local file instead of the database.

Table 3–8: Client Database Connection Parameters *(2 of 4)*

Description	Syntax	Purpose
Communications Parameter File	-cp { *com-parms* | *filename* }	Override preset connection parameters for auto-connect databases.
Cursor Size	-cs *n*	Change the maximum number of index levels.
Physical Database Name	-db *physical-dbname*	Identify one or more databases to connect to at startup.
DataService	-DataService *data-service*	Connect through a NameServer to an ODBC or ORACLE DataServer.
Direct I/O [1, 2]	-directio	Open all files in unbuffered mode.
Database Type	-dt *db-type*	Identify the database type.
Force Access	-F	Allow access to a damaged database. Call Technical Support before using -F.
Host Name	**TCP:** -H *host-name* **SNA:** PLU= plu-*name*	Identify a remote host.
Number of Databases	-h *n*	Specify the maximum number of databases that can be connected.
Hash Table Entries [1]	-hash *n*	Specify the number of hash table entries for the buffer pool.
No Crash Protection [1]	-i	Run Progress without database integrity or recovery.

Table 3–8: Client Database Connection Parameters *(3 of 4)*

Description	Syntax	Purpose
AS/400 Ignore Stamp	-is	Stop a client from comparing time stamps when connecting to an AS/400 database.
Logical Database Name	-ld *logical-dbname*	Assign the logical database name.
Delayed BI File Write [1, 2]	-Mf *n*	Delay writing the last before-image (BI) file records.
Network Type	-N *network-type*	Specify the network communications protocol.
Password [3]	-P *password*	Specify a user password.
Fast Schema Change	-populate	Turn off Fast schema change.
Non-reliable I/O [1]	-r	Enable buffered I/O to the before-image file.
Read-only Media	-RO	Open a database for read-only access.
Service Name	-S { *servicename* | *port-number* }	Identify the service and port number to use on the host machine when connecting to a broker process.
AS/400 Server Program Name	-Sn	Specify that the Server program on an AS/400 start with a Progress connection request (SNA only).

Table 3–8: Client Database Connection Parameters *(4 of 4)*

Description	Syntax	Purpose
Trigger Location	-trig { *dir-name* | *lib-name* }	Identify the directory or library containing the 4GL triggers for a database.
User ID [3]	-U *userid*	Specify the user ID.

[1] Single-user database connections only; otherwise, use only for server startup.

[2] UNIX, NT only.

[3] Different meaning for non-Progress databases.

3.4 Progress Database Server Parameters (DBS)

The Progress Database Server parameters are organized alphabetically by syntax in the following subcategories:

- Database Server Performance
- Database Server-type
- Database Server Internationalization
- Database Server Statistics
- Database Server Network

3.4.1 Database Server Performance Parameters

Use the parameters listed in Table 3–9 to optimize server performance.

Table 3–9: Server Performance Parameters *(1 of 2)*

Parameter	Syntax	Purpose
After-image Buffers	-aibufs *n*	Specify the number of after-image buffers when running AIW.
After-image Stall	-aistall	Suspend database activity when an empty after-image (AI) file is unavailable.
Blocks in Database Buffers	-B *n*	Specify the number of blocks in the database buffers.
Maximum Private Buffers per Connection	-Bpmax	Control the number of private buffers a connection can request through the Private Read-only Buffers (`-Bp`) parameter.
Before-image Buffers	-bibufs *n*	Specify the number of before-image buffers when running BIW.
Threshold Stall	-bistall	Send a message to the log file when the recovery log threshold is reached. Use with -bithold.
Recovery Log Threshold	-bithold *n*	Specify the maximum size of the recovery log files.
Direct I/O[1]	-directio	Open all files in unbuffered mode.
Before-image Cluster Age	-G *n*	Specify the number of seconds before Progress reuses a before-image cluster.
Group Delay	-groupdelay *n*	Specifies the number of milliseconds a transaction waits before committing.
Hash Table Entries	-hash	Specify the number of hash table entries for the buffer pool.
No Crash Protection	-i	Run Progress without using database integrity or recovery.

Table 3–9: Server Performance Parameters *(2 of 2)*

Parameter	Syntax	Purpose
Lock Table Entries	-L *n*	Specify the number of entries in the record locking table.
Delayed BI File Write[1]	-Mf *n*	Delay writing the last before-image (BI) file records.
VLM Page Table Entry Optimization[2]	-Mpte	Allocate shared memory in multiples of 8 MB for VLM64 support.
Shared-memory Overflow Size[1]	-Mxs *n*	Replace the default value of the shared-memory overflow area.
Number of Users	-n *n*	Specify the maximum number of users connected to the database.
Pin Shared Memory	-pinshm	Prevent the database engine from swapping shared memory contents to disk.
Semaphore Sets[3]	-semsets *n*	Change the number of semaphore sets available to a broker.
Spin Lock Retries[1]	-spin *n*	Specify the number of times a process tries to acquire a latch before pausing.

1 UNIX, NT only.

2 Digital UNIX only.

3 UNIX only.

3.4.2 Database Server-type Parameters

Use the parameters listed in Table 3–10 to start a particular type of server.

Table 3–10: Server-type Parameters

Parameter	Syntax	Purpose
Auto Server	-m1	Start an auto server. Used internally by the Database broker.
Manual Server[1]	-m2	Manually start a remote user server after you start a broker.
Secondary Login Broker[1]	-m3	Start a secondary broker.
Type of Server to Start	-ServerType	Specify whether to start a 4GL server, a SQL server, or both.

[1] UNIX, NT only.

3.4.3 Database Server Internationalization Parameters

Use the parameters listed in Table 3–11 to control the format in which data appears.

Table 3–11: Server Internationalization Parameters *(1 of 2)*

Parameter	Syntax	Purpose
Conversion Map	-convmap *filename*	Identify the conversion map file.
Case Table	-cpcase *tablename*	Identify the case table that establishes case rules for the code page.
Collation Table	-cpcoll *tablename*	Identify a collation table to use with the code page.
Internal Code Page	-cpinternal *codepage*	Identify the code page that Progress uses in memory.
Log File Code Page	-cplog *codepage*	Identify the code page used for writing messages to the log file.

Table 3–11: Server Internationalization Parameters *(2 of 2)*

Parameter	Syntax	Purpose
Print Code Page	-cpprint *codepage*	Identify the code page used for printer output.
R-code in Code Page	-cprcodein *codepage*	Identify the code page used for reading r-code text segments.
Stream Code Page	-cpstream *codepage*	Identify the code page used for stream I/O.
Terminal Code Page	-cpterm *codepage*	Identify the code page for character terminals.

3.4.4 Database Server Statistics Collection Parameters

Use the parameters listed in Table 3–12 to collect statistics for table and index access.

Table 3–12: Server Statistics Collection Parameters *(1 of 2)*

Parameter	Syntax	Purpose
Base Index	-baseindex *n*	Specify a range of indexes for which you want to collect statistics. Use with -indexrangesize.
Base Table	-basetable *n*	Specify a starting table number in a range of tables for which you want to track access statistics. Use with -tablerangesize.
Index Range Size	-indexrangesize *n*	Specify the number of indexes to track for access statistics.
License Usage Reporting Interval[1]	-rptint *n*	Define the polling interval that a server uses to gather license usage statistics.

Table 3–12: Server Statistics Collection Parameters *(2 of 2)*

Parameter	Syntax	Purpose
Table Range Size	-tablerangesize *n*	Specify the number of tables for which you want to collect statistics.
User Count	-usrcount *n*	Specify the number of users allowed to connect to a database before license usage reporting begins.

[1] UNIX, NT only.

3.4.5 Database Server Network Parameters

Use the parameters listed in Table 3–13 to supply Progress with necessary network information.

Table 3–13: Server Network Parameters *(1 of 2)*

Parameter	Syntax	Purpose
AdminServer Port	-adminport { *service-name* | *port* }	Connect a servergroup and an AdminServer.
SQL-92 Server Java Classpath	-classpath *pathname*	Identify the Java classpath to use when starting an SQL server.
Host Name	**TCP:** -H *host-name* **SNA:** -H PLU= plu-*name*	Specify a remote host.
Maximum Clients Per Server[1]	-Ma *n*	Specify the maximum number of remote users per database server.
Maximum Dynamic Server[1]	-maxport *n*	Specify the highest accessible port number in a specified range.
Minimum Clients per Server[1]	-Mi *n*	Specify the number of remote users on a server before a broker starts another server.

Table 3–13: Server Network Parameters *(2 of 2)*

Parameter	Syntax	Purpose
Minimum Dynamic Server[1]	-minport *n*	Specify the lowest accessible port number in a specified range.
Maximum Servers[1]	-Mn *n*	Specify the maximum number of remote client servers that a broker can start.
Servers Per Protocol[1]	-Mp *n*	Specify the maximum number of servers to serve remote users for a protocol.
Maximum Servers Per Broker[1]	-Mpb *n*	Specify the maximum number of servers that multiple brokers can start to serve remote users for a protocol.
Network Type	-N *network-type*	Identify the network communications protocol.
Configuration Properties File	-properties *filename*	Identify the properties file an AdminServer uses when starting a database server or servergroup.
Service Name	-S { *service-name* | *port-number* }	Specify the service or port number to be used by a broker process.
Server Group	-servergroup *name*	Identify a logical collection of server processes to start.

[1] UNIX, NT only.

3.5 DataServer Parameters (DS)

Use the parameters listed in Table 3–14 when starting an ODBC, ORACLE, or Progress/400 DataServer. The parameters are organized alphabetically by syntax.

Table 3–14: DataServer Parameters *(1 of 2)*

Parameter	Syntax	Purpose
Index Cursors	-c *n*	Specify the number of index place holders or *cursors*.
Internal Code Page	-cpinternal *codepage*	Identify the code page that Progress uses in memory.
Log File Code Page	-cplog *codepage*	Identify the code page used for writing messages to the log file.
Print Code Page	-cpprint *codepage*	Identify the code page used for printer output.
R-code in Code Page	-cprcodein *codepage*	Identify the code page used for reading r-code text segments.
Stream Code Page	-cpstream *codepage*	Identify the code page used for stream I/O.
Terminal Code Page	-cpterm *codepage*	Identify the code page for character terminals.
DataServer	-Dsrv *keyword* (, *value*) [, *keyword2* (, *value2*) ...]	Identify keywords as parameters for the ODBC, ORACLE, or Progress/400 DataServer.
Database Type	-dt *db-type*	Identify the database type.
Host Name	**TCP:** -H *host-name* **SNA:** -H PLU= plu-*name*	Specify a remote host.
AS/400 Ignore Stamp	-is	Stop a client from comparing time stamps when connecting to an AS/400 database.
Index Hint	-noindexhint	Stop an ORACLE DataServer from providing index hints to the ORACLE DBMS.

Table 3–14: DataServer Parameters *(2 of 2)*

Parameter	Syntax	Purpose
Server Join	-nojoinbysqldb	Make a client evaluate and perform queries that have joins.
SELECT Pass Through Disable	-noSQLbyserver	Turn off SELECT pass Through for OBDC and ORACLE DataServers.
AS/400 Server Program Name	-Sn	Start a Server program on an AS/400 with a Progress connection request (SNA only).

3.6 Open Interface Driver Parameters (OID)

Use the parameters listed in Table 3–15 with ESQL-89 clients. The parameters are organized alphabetically by syntax.

Table 3–15: ESQL-89 Client Parameters *(1 of 2)*

Parameter	Syntax	Purpose
Case Table	-cpcase *tablename*	Identify the case table that establishes case rules for the code page.
Collation Table	-cpcoll *tablename*	Identify a collation table to use with the code page.
Internal Code Page	-cpinternal [*codepage*]	Identify the code page that Progress uses in memory.
Log File Code Page	-cplog *codepage*	Identify the code page used for writing messages to the log file.
Print Code Page	-cpprint *codepage*	Identify the code page used for printer output.
R-code in Code Page	-cprcodein *codepage*	Identify the code page used for reading r-code text segments.

Table 3–15: ESQL-89 Client Parameters *(2 of 2)*

Parameter	Syntax	Purpose
Stream Code Page	-cpstream *codepage*	Identify the code page used for stream I/O.
Terminal Code Page	-cpterm *codepage*	Identify the code page for character terminals.
ESQL Log	-esqllog	Send messages to the database log as well as the message queue.
ESQL No Padding	-esqlnopad	Enable SQL to recognize hyphens and minus signs correctly.
No Lock	-NL	Make all record retrieval statements default to NO-LOCK record access (Version 7.3B and later).
ESQL PROPATH	-pp *directory*	Add a new directory to the existing PROPATH.
ANSI SQL	-Q	Enforce strict ANSI SQL conformance when processing Progress/SQL statements.
ANSI SQL Client	-Q2	Make clients conform to ANSI SQL89 rules for serializing transactions.
Open Interface Driver Connection (OID)	-SV *parameter-list*	Identify a parameter list that specifies the connection between an ESQL small client or ODBC Driver and OIB/OID.

4

Startup Parameter Descriptions

This chapter describes the Progress startup parameters in detail, in alphabetical order by syntax. Each section begins with a table that shows the syntax of the parameter and other information about it. The Use With column indicates the usage categories of the parameter. The usage categories are:

- Client Connection (CC), to use when connecting to a specific database
- Client Session (CS), to use when starting any client session
- Database Server (DBS), to use when starting a server or broker
- DataServer (DS), to use when starting non-Progress DataServers
- Open Interface Driver (OID), to use when starting the Open Interface Driver

For a quick reference to the parameters, see Chapter 3, “Startup Parameter Usage Categories.”

Single-user Mode (-1)

Operating System and Syntax	UNIX Windows	-1		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC	–	–	–	–

Use Single-user Mode (-1) with a Progress executable module to start Progress in single-user mode. In this mode, only one user can access the database.

The PRO Command invokes the Progress executable with -1 by default, but only for the first database named on the command line. Each additional database must already have a server running in multi-user mode, unless a -1 is added to its parameter list. In the following example, db1 and db3 are connected in single-user mode; db2 is connected in multi-user mode:

```
pro db1 -db db2 -db db3 -1
```

AdminServer Port (-adminport)

Operating System and Syntax	UNIX Windows	-adminport *port*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	–	–	–	–

service-name

The name of the service to be used by the AdminServer.

port

The port number the AdminServer uses to communicate with servergroups.

Use AdminServer Port (-adminport) to establish communication between a servergroup and an AdminServer. The AdminServer uses this parameter internally. The default is none, and the server does not attempt to connect to an AdminServer. You do not usually use this parameter directly. The -adminport setting must match the -admin setting specified when the AdminServer was started.

After-image Buffers (-aibufs)

Operating System and Syntax	UNIX Windows	-aibufs *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	–	1	–	5

n

The number of after-image buffers to use.

Use After-image buffer (-aibufs) to specify the number of after-image buffers. This parameter is useful only when running the after-image writer (AIW) because the AIW writes the filled after-image buffers to disk, making the buffers available to other client and server processes. Progress Software recommends you set -aibufs to a value of 1.5 times the value of the Before-image Buffers (-bibufs) parameter, or a minimum of 5.

Without the AIW writing the buffers, any gain from increasing the number of buffers is negligible.

After-image Stall (-aistall)

Operating System and Syntax	UNIX Windows	-aistall		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	–	–	–	–

Use After-image Stall (-aistall) to suspend database activity if all AI files are filled. Then wait until the next AI file is backed up and marked empty and send this message to the database log file:

```
Can't switch to after-image extent filename it is full. (3775)
Backup ai extent and mark it as empty (3776)
```

This way you can empty an AI file instead of shutting down the database.

When using after-image (AI) files, you monitor the status of the files to ensure that you do not attempt to reuse an unavailable file. If all the AI files are filled and Progress cannot switch to an empty one, by default the database shuts down. See the *Progress Database Administration Guide and Reference* for information on using the after imaging feature.

Async Queue Size (-asyncqueuesize)

Operating System and Syntax	UNIX Windows	-asyncqueuesize *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	2,147,483,647	1	–	–

n

The total number of bytes allowed for all asynchronous request buffers allocated for a client.

Use Async Queue Size (-asyncqueuesize) to specify the total number of bytes allowed for all buffers allocated when asynchronous requests are queued on behalf of a client. This number determines how many buffers to allocate for each client connection. Each buffer is at most 8KB. If you do not specify -asyncqueuesize, the default size for all buffers is 64KB.

As each client connects, request buffers are generated and the size of each buffer determined. Both send and receive requests can be queued on a client depending how the program is implemented. When the buffers are filled, additional queued requests are stored in a temporary table associated with that client session. If you set the value of -asyncqueuesize to 0, all requests are placed in the temporary table.

Performance may vary based upon whether requests are stored in memory or in the temporary table. So you may want to choose this value based upon how much data you expect to flow between the client and the AppServer for asynchronous requests.

NOTE: You can use -asyncqueuesize only when a client making asynchronous requests is connecting to an AppServer. The client may be an AppServer.

The following example shows how to start a multi-user mode Progress client session using -asyncqueuesize:

```
mpro -asyncqueuesize 135000
```

Blocks in Database Buffers (-B)

Operating System and Syntax	UNIX Windows	-B *n*		
Use With	**Maximum Value**[1]	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC, DBS	System dependent[1]	10	20	(8 * *users*) [2]

[1] Limited by available memory.

[2] The *users* value is specified by the Number of Users (-n) parameter.

n

The number of blocks in the database buffers.

Use Database Buffers (-B) to specify the number of blocks in the database buffers. The optimum value depends on your application. See the *Progress Database Administration Guide and Reference* for more information.

Batch (-b)

Operating System and Syntax	UNIX Windows	-b		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

Use Batch (-b) to initiate a batch session, with no terminal interaction. When procedures run in batch mode, Progress does not display any messages produced by those procedures on the terminal screen.

ADE r-code Location (-baseADE)

Operating System and Syntax	UNIX Windows	-baseADE <*directory*>		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	-	-	-	-

<*directory*>

Identifies where the client should look for the ADE r-code.

It is now possible to have different versions of ADE r-code on the same machine at the same time. As a result, you need a way to specify the ADE r-code directory to the client so that the client can add this directory and all of the procedure libraries contained in the directory to the PROPATH. The new startup parameter -baseADE lets you specify the location of the ADE r-code directory.

At session startup, Progress automatically adds $DLC/gui (or $DLC/tty for character mode clients) and all of the procedure libraries in $DLC/gui (or $DLC/tty) to PROPATH.

When you specify -baseADE, the $DLC/gui (or $DLC/tty) directory and the procedure libraries in $DLC/gui (or $DLC/tty) are not added to PROPATH. Instead, the directory specified using the -baseADE startup parameter is added to PROPATH followed by all of the procedure libraries in the directory. For example, if you use -baseADE to specify the following:

```
-baseADE c:\maint\gui
```

And c:\maint\gui contained the following procedure libraries: adecomm.pl, adeuib.pl, and protools.pl

The PROPATH looks like the following:

```
...,c:\maint\gui,c:\maint\gui\adecomm.pl,c:\maint\
gui\adeuib.pl,c:\maint\gui\protools.pl,...
```

If $DLC/gui (or $DLC/tty) or any of the procedure libraries from the directory are already part of PROPATH (environment variable, .ini file, or registry setting), Progress does not remove them.

Progress allows you to specify an empty string for <*directory*>, `-baseADE ""`. In this situation, Progress does not add `$DLC/gui` (or `$DLC/tty`) or the procedure libraries in `$DLC/gui` (or `$DLC/tty`) to PROPATH.

You can also specify the ADE r-code location with the BASE-ADE attribute of the SESSION system handle. For more information about this SESSION attribute, see *Progress Version 9 Product Update Bulletin*.

Base Index (-baseindex)

Operating System and Syntax	UNIX Windows	-baseindex *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	–	–	–	–

n

The starting index number in the range of indexes for which you want to track access statistics.

Use Base Index (-baseindex) with Index Range Size (-indexrangesize) to specify the range of indexes for which you want to collect statistics. Access to the statistics is handled through the Virtual System Tables (VSTs). Index statistics are stored in the _IndexStat VST. See the *Progress Database Administration Guide and Reference* for more information on virtual system tables. To obtain index numbers, use the following 4GL code:

```
FOR EACH _file:
     DISPLAY _file.file-filename.
     FOR EACH _index WHERE index.file-recid = RECID(file):
           DISPLAY _idx-num _index-name.
     END.
END.
```

This results in the following output:

```
File-Name
filename

_idx-num Index-Name
     n1   index name1
     n2   index name2
     n3   index name3
```

Registry Base Key (-basekey)

Operating System and Syntax	Windows	-basekey *name*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

name

The name of the registry basekey.

Use Registry Base Key (-basekey) to identify the registry basekey in which to look for environment information.

You can force Progress to bypass the registry search by specifying INI. Valid values are:

- HKEY_CURRENT_USER
- HKEY_CLASSES_ROOT
- HKEY_LOCAL_MACHINE
- HKEY_USERS
- HKEY_CURRENT_CONFIG
- HKEY_DYN_DATA
- INI

NOTE: You can only use the -basekey startup parameter at the command line. It is ignored when used in a parameter (.pf) file.

See the *Progress Client Deployment Guide* for more information on using the -basekey startup parameter.

Base Table (-basetable)

Operating System and Syntax	UNIX Windows	-basetable *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	–	–	–	–

n

The starting table number in the range of tables for which you want to track access statistics.

Use Base Table (-basetable) with Table Range Size (-tablerangesize) to specify the range of tables for which you want to collect statistics. Access to the statistics is handled through the Virtual System Tables (VSTs). Table statistics are stored in the _TableStat VST. See the *Progress Database Administration Guide and Reference* for more information on virtual system tables. To obtain table numbers, use the following 4GL code:

```
FOR EACH _file:
     DISPLAY _file-num _file-filename.
END.
```

This results in the following output:

```
_File-Number  File-Name
     n1   table name1
     n2   table name2
      n3   table name3
```

Before-image Buffers (-bibufs)

Operating System and Syntax	UNIX Windows	-bibufs *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	–	3	–	5

n

The number of before-image buffers.

Use Before-image Buffers (-bibufs) to specify the number of before-image buffers. This parameter is useful only when running the before-image writer (BIW). The BIW continually writes the filled before-image buffers to disk, making the buffers available to other client and server processes. Without a BIW writing the buffers, any gain from increasing the number of buffers is negligible.

See the *Progress Database Administration Guide and Reference* for more information on setting this parameter.

Threshold Stall (-bistall)

Operating System and Syntax	UNIX Windows	-bistall		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	–	–	–	–

Use Threshold Stall (-bistall) with Recovery Log Threshold (-bithold) to quiet the database when the recovery log threshold is reached, without performing an emergency shutdown. When you use -bistall, a message is added to the database log (.lg) file stating that the threshold stall is enabled.

Recovery Log Threshold (-bithold)

Operating System and Syntax	UNIX Windows	`-bithold n`		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	System dependent[1]	System dependent[1]	–	–

[1] Limited by available disk space.

n

An integer specifying the threshold, in MB.

Use Recovery Log Threshold (-bithold) to set the maximum size to which recovery log files can grow. The recommended threshold is between 3% and 100% of the largest possible recovery log file size, rounded to the nearest cluster boundary. If the threshold is set above 1000MB, Progress issues a warning message to the display and the database log (`.lg`) file. Once the threshold is reached, the database performs an emergency shutdown. See the *Progress Database Administration Guide and Reference* for more information about recovery logs.

Private Read-only Buffers (-Bp)

Operating System and Syntax	UNIX Windows	-Bp *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC	64	1	–	–

n

An integer value greater than 1 but less than 64.

Use Private Read-only Buffer (-Bp) to request a number of private read-only buffers. The request remains active for the entire session unless it is changed or disabled by an application. Each user of private read-only buffers reduces the number of public buffers, which is set with Blocks in Database Buffers (-B).

NOTE: The total number of private read-only buffers for all simultaneous users is limited to 25% of the total blocks in database buffers.

Maximum Private Buffers per Connection (-Bpmax)

Operating System and Syntax	UNIX Windows	-Bpmax *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	25% of -B	1	64	64

n

An integer specifying the maximum number of private buffers.

The Maximum Private Buffers per Connection (-Bpmax) parameter is a broker setting which controls the number of private buffers a connection can request through the Private Read-only Buffers (-Bp) parameter.

The number of blocks in the database buffers is controlled by the Blocks in Database Buffers (-B) parameter. Up to 25% of the blocks created may be allocated for use as private buffers. When a client connects to a Progress database, the connection requests -Bp *n* private buffers. Prior to 9.1C, the maximum value of -Bp was 64. With the new -Bpmax parameter, the broker may override this limit by allowing the value of -Bp to be up to 25% of -B.

NOTE: Each use of a private read-only buffer reduces the number of public buffers. The total number of private buffers for all simultaneous users is limited to 25% of the total blocks in database buffers.

Bleeding Record Lock (-brl)

Operating System and Syntax	UNIX Windows	-brl		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC	–	–	–	–

Use Bleeding Record Lock (-brl) to let Progress hold a share lock on a record used by more than one buffer, even after the buffer using the record releases it.

Buffer Size for Temporary Tables (-Bt)

Operating System and Syntax	UNIX Windows	-Bt *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	50,000	10	–	–

n

An integer that specifies the number of blocks in the buffer for temporary tables.

Use Buffer Size for Temporary Tables (-Bt) to specify the size of the buffer pool for blocks/records from the temporary table. Progress uses the value you specify for temporary databases used during the session the same way it uses Blocks in Database Buffers (-B) for permanent databases. The default value is 10.

See the *Progress Programming Handbook* for more information on temporary tables.

Index Cursors (-c)

Operating System and Syntax	UNIX Windows	-c *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DS	–	–	–	–

n

The number of index cursors allowed.

Use Index Cursors (-c) to specify the number of index place holders, or *cursors*, for a Progress session. Each active FOR EACH and FIND NEXT loop uses one index cursor for each index Progress is using. A cursor takes 64 bytes. If you need more index cursors, Progress displays the following error message:

```
SYSTEM ERROR: Too few index cursors. Increase -c parameter.
```

NOTE: The -c parameter has a different meaning when you connect to an ORACLE database. For more information, see the *Progress DataServer for ORACLE Guide*.

Schema Cache File (-cache)

Operating System and Syntax	UNIX Windows	-cache *filename*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC	–	–	–	–

filename

The pathname of a binary schema cache file.

Use Schema Cache File (-cache) to read the database schema from a local file instead of the database. You must have previously built the schema cache and stored it as a binary file.

To perform database activities, the Progress client keeps a copy of the database schema called the *schema cache* in memory. By default, Progress creates the schema cache by reading the database schema stored in the database file. The time required to read the schema is usually minimal. However, under the following conditions, the time required to read the schema might be unacceptable:

- If the client connects to the database over a wide-area network (WAN).
- When a large number of clients connect to a database simultaneously. For example, after a database shutdown or crash.

Connection time depends on several factors, including schema size.

To reduce connection time, Progress lets you store the schema cache as a binary file, called a *schema cache file*, on a local disk. The client can then read the schema directly from the schema cache file.

To write the schema cache file, you build the desired schema cache and save it to a binary file using the Progress statement SAVE CACHE. The schema cache file is portable across systems, so you can create the file once and distribute it across a heterogeneous network of systems. For information on building and saving the schema cache file, see the *Progress Programming Handbook*.

If you specify Schema Cache File (-cache) when you connect to a database and the local schema is valid, Progress reads the schema from the local file instead of from the database. The schema cache is valid if the time stamp of the schema cache file matches the time stamp in the database master block. If the time stamps do not match, or for some reason Progress cannot read the file, Progress issues a warning message and reads the schema from the database.

NOTE: If you are generating the local binary schema cache, do not connect to the database using Trigger Location (-trig) and Schema Cache File (-cache) together.

Check Double-byte Enabled (-checkdbe)

Operating System and Syntax	UNIX Windows	-checkdbe		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

Use Check Double-byte Enabled (-checkdbe) when you modify a Progress application to support double-byte characters or extended character data. The -checkdbe parameter specifies that Progress generates compile-time warning messages whenever it finds a LENGTH function, SUBSTRING function, SUBSTRING statement, or OVERLAY statement without a CHARACTER, COLUMN, RAW, or FIXED option.

SQL-92 Server Java Classpath (-classpath)

Operating System and Syntax	UNIX Windows	-classpath *pathname*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	–	–	–	–

pathname

Specifies the pathname of the classpath.

Use SQL-92 Classpath (-classpath) to identify the Java classpath to use when starting an SQL server. SQL-92 database brokers use this parameter when launching the Java Virtual Machine (JVM) to execute stored procedures. The default is to use the current environment variable CLASSPATH setting. You do not use this parameter directly.

Client Logging (-clientlog)

Operating System and Syntax	UNIX Windows	-clientlog *filename*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	-	-	-	-

filename

Name of log file used for messages and 4GL stack trace information.

If the filename you supply is a relative pathname, then a file is accessed relative to the current working directory. If the filename is an absolute pathname, then the specified file is accessed.

Client logging allows an application to automatically write all error and warning messages to the specified log file. If you use the `-clientlog` parameter at startup, all messages displayed on the screen in an alert box are also written to the log file you specified with `-clientlog`.

When you use the `-clientlog` startup parameter, and you also specify the `-debugalert` startup parameter or set SESSION:DEBUG-ALERT to yes, the log includes a 4GL stack trace for each log entry.

On connect and disconnect from a database, an entry appears in the client log file identifying the database and userid (for multi-user connections). This information can be used to relate client log messages with database log messages.

If an error message is diverted to the ERROR-STATUS system handle, and client logging is enabled, then no information is written to the log file.

In a non-interactive session, the application is configured so that the output device is associated with a file (or another device). In this configuration, when a Progress statement encounters an error, it writes the error to the output device. If client logging is enabled, then this message is also be written to the specified log file.

You can use the MESSAGE statement with the VIEW-AS ALERT-BOX to write application specific information to the screen. All information written to the screen using the MESSAGE VIEW-AS ALERT-BOX is also written to the log file.

When -debug-alert is on and an error occurs in a batch session for:

- A 4GL batch client, Progress records the 4GL stack trace information in the log file specified with -clientlog. In this case, Progress lists the stack trace information in reverse order. That is, the procedure that generated the error is written first, and the root procedure is written last.
- An AppServer client or a WebSpeed agent, Progress records the 4GL stack trace information in their respective log files, also in reverse order. If you view one of these log files in the Progress Explorer, it appears in top-down order since Explorer displays log files in reverse order.

The code page used for the log file is cpinternal and no conversions are performed for these log files.

Conversion Map (-convmap)

Operating System and Syntax	UNIX Windows	-convmap *filename*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS, DBS	–	–	–	–

filename

Pathname of your CONVMAP file.

Use Conversion Map (-convmap) to identify the CONVMAP file to use for code page conversions, collation orders, and case conversions. By default, Progress uses the convmap.cp file in the DLC directory. You can create a CONVMAP file by using the PROUTIL utility with the CODEPAGE-COMPILER qualifier. See the *Progress Internationalization Guide* for more information on CONVMAP files.

Communications Parameter File (-cp)

Operating System and Syntax	UNIX Windows	-cp { *com-parms* | *filename* }		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC	–	–	–	–

com-parms, *filename*

The database connection parameters or parameter file.

Use Communications Parameter File (-cp) to override preset connection parameters for auto-connect databases. Communications parameters supplied with the -cp parameter override any connection information stored in the Progress metaschema field _Db-comm in the _Db file record for the database.

Case Table (-cpcase)

Operating System and Syntax	UNIX Windows	-cpcase *tablename*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS, DBS, OID	–	–	Basic	Basic

tablename

Name of a case table in the `convmap.cp` file.

Use Case Table (-cpcase) to specify the case table. This table establishes case rules for the code page that Progress uses in memory. The code page is specified by the Internal Code Page (-cpinternal) parameter. The case rules are used by the CAPS and LC functions. Also, in a character field format, you can use an exclamation point (!) to tell Progress to convert all characters to uppercase during input.

To retrieve the value of this startup parameter at runtime, use the SESSION System handle.

Collation Table (-cpcoll)

Operating System and Syntax	UNIX Windows	-cpcoll *tablename*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS, DBS, OID	–	–	Basic	Basic

tablename

Name of a collation table within the `convmap.cp` file.

Use Collation Table (-cpcoll) to identify a collation table that Progress uses with the code page in memory. The code page is specified by the Internal Code Page (-cpinternal) parameter.

Progress uses the collation rules that you specify to compare characters and sort records if a BY clause cannot be satisfied by an index. The collation rules specified with the -cpcoll parameter take precedence over the collation specified for any database Progress accesses during the session, except when Progress uses or modifies pre-existing indexes. When you update or rebuild a database's indexes, Progress uses the collation rules originally defined for that database.

If you do not use -cpcoll, Progress uses the language collation rules defined for the first database on the command line. If you do not specify a database on the command line, Progress uses the collation rules with the name "basic."

To retrieve the value of this startup parameter at runtime, use the SESSION System handle.

See the *Progress Internationalization Guide* for more information on collation tables.

Internal Code Page (-cpinternal)

Operating System and Syntax	UNIX Windows	-cpinternal *codepage*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS, DBS, DS, OID	–	–	iso8859-1[1]	iso8859-1[1]

[1] If you are using Progress/400, the default code page is ibm037.

codepage

Name of the code page that Progress uses in memory.

Use Internal Code Page (-cpinternal) to identify the code page that Progress uses in memory and for graphical clients. For graphical clients, the -cpinternal code page should be the same code page that the operating system uses. If you do not use -cpinternal, the iso8859-1 code page is used by default.

NOTE: Do not use a 7-bit table with -cpinternal. Use 7-bit tables for converting data from a 7-bit terminal to another code page only. Do not use them for character conversion in memory or for the database.

To retrieve the value of this startup parameter at runtime, use the CPINTERNAL attribute of the SESSION System handle.

Log File Code Page (-cplog)

Operating System and Syntax	UNIX Windows	-cplog *codepage*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS, DBS, DS, OID	–	–	-cpstream	-cpstream

codepage

Name of the code page for messages written to the log file.

Use Log File Code Page (-cplog) to identify the code page that Progress uses to write messages to the log (.lg) file. If you do not specify a value, the default is the code page specified by Stream Code Page (-cpstream).

To retrieve value of this startup parameter at runtime, use the CPLOG attribute of the SESSION System handle.

Print Code Page (-cpprint)

Operating System and Syntax	UNIX Windows	-cpprint *codepage*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS, DBS, DS. OID	–	–	-cpstream	-cpstream

codepage

Name of the code page used for printer output.

Use Print Code Page (-cpprint) to identify the code page Progress uses when it prints. When you print a file, the code page specified by -cpprint overrides the code page specified by Stream Code Page (-cpstream).

To retrieve the value of this startup parameter at runtime, use the SESSION System handle.

R-code in Code Page (-cprcodein)

Operating System and Syntax	UNIX Windows	-cprcodein *codepage*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS, DBS, DS, OID	–	–	-cpinternal	-cpinternal

codepage

Name of the code page for reading r-code text segments.

Use R-code In Code Page (-cprcodein) to read the r-code text segments, as if they were written in the code page specified by -cprcodein, and convert them to the Internal Code Page (-cpinternal) code page. Usually when Progress reads r-code, it converts text segments to the code page specified by Internal Code Page (-cpinternal).

CAUTION: This parameter is for use during very rare situations and in general should not be used. Progress reads text segments as if they are written in the code page specified by -cprcodein, even if the text segments were written with a different code page. For example, if you use the following startup parameters and run a .r file written with code page IBM850, Progress converts the text segments from ISO8859-1 to ibm861. This can produce incorrect results, if the .r file was correctly labeled internally as IBM850:

```
-cprcodein ISO8859-1 -cpinternal ibm861
```

To retrieve the value of this startup parameter at runtime, use the SESSION System handle. To determine the code page of an r-code file, use the RCODE-INFO handle.

R-code out Code Page (-cprcodeout)

Operating System and Syntax	UNIX Windows	-cprcodeout *codepage*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	-cpinternal	-cpinternal

codepage

Name of the code page for writing r-code text segments.

Use R-code out Code Page (-cprcodeout) to direct Progress to use the code page you specify to mark and convert text segments (any translatable text) when it writes r-code. If you do not specify cprcodeout, Progress uses the code page you specify with Internal Code Page (-cpinternal) to write r-code.

Typically, you do not need to specify this parameter since Progress converts the text segment to the -cpinternal code page when reading r-code. If the code page of the intended user is known, however, you can use this parameter to provide r-code in the user's code page. Performance savings are not significant.

To retrieve the value of this startup parameter at runtime, use the SESSION System handle. To determine the code page of an r-code file, use the RCODE-INFO handle.

Stream Code Page (-cpstream)

Operating System and Syntax	UNIX Windows	-cpstream *codepage*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS, DBS, DS, OID	–	–	ibm850	ibm850

codepage

Name of the code page for stream I/O.

Use Stream Code Page (-cpstream) to identify the code page Progress uses for stream I/O. Character terminals use the code page you specify for -cpstream unless you also specify a value for Terminal Code Page (-cpterm), Print Code Page (-cpprint), or Log File Code Page (-cplog).

Stream I/O consists of the following elements:

- Terminals (includes character terminals and DOS Protected mode, but does not include graphical interfaces or the Windows character interface)
- Data (.d) files
- READ-FILE, WRITE-FILE, and INSERT-FILE methods for the EDITOR widget
- INPUT FROM and OUTPUT TO statements
- All compilable files (`.p`, `.w`, `.i`, and so on.)
- Compiler-generated LISTING, XREF, and PREPROCESS files

NOTE: Do not use a 7-bit table with -cpstream. Use 7-bit tables for converting data from a 7-bit terminal to another code page only. Do not use them for character conversion in memory or for the database.

To retrieve the value of this startup parameter at runtime, use the SESSION System handle. To determine the code page of an r-code file, use the RCODE-INFO handle.

Terminal Code Page (-cpterm)

Operating System and Syntax	UNIX Windows	`-cpterm` *codepage*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS, DBS, DS, OID	–	–	-cpstream	-cpstream

codepage

Name of the code page for character terminals.

Use Terminal Code Page (-cpterm) to identify the code page of your character terminals. This parameter allows you to specify a different code page for character terminals than used by the rest of stream I/O, which is set by Stream Code Page (-cpstream).

NOTE: You can use a 7-bit table with -cpterm.

To retrieve the value of this startup parameter at runtime, use the SESSION System handle. To determine the code page of an r-code file, use the RCODE-INFO handle.

Cursor Size (-cs)

Operating System and Syntax	UNIX Windows	-cs *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC	256	1	6	6

n

The cursor size, which is the number of index levels.

Use Cursor Size (-cs) to change the maximum number of index levels. You seldom need to use this parameter.

Progress indexes have a tree structure. Even large indexes typically are no deeper than three or four levels. Index cursors, which act as place holders during file access, have size 6 by default; that is, they support up to five levels of indirect access. Therefore, few users ever require the Cursor Size (-cs) parameter. However, in rare cases involving large index keys, increase -cs in response to the following error message:

```
Index index-number has at least n levels. Increase -cs parameter.
```

Increasing -cs by 2 should be sufficient.

Directory Size (-D)

Operating System and Syntax	UNIX Windows	-D *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	500	5	100	100

n

The number of compiled procedure directory entries.

Use Directory Size (-D) to change the number of compiled procedure directory entries. Each compiled procedure executed during a Progress session requires a directory entry. A compiled procedure can be a session-compiled version of a procedure or a precompiled r-code version.

When Progress creates a session-compiled version of a procedure and there is no available space in the directory, it discards the oldest inactive compilation of a procedure. The next time the discarded procedure is run, Progress must recompile it if it was a session compile, or reopen and reread it if it was a precompiled r-code version.

The -D limit is a soft limit. If your application needs to exceed the limit, Progress automatically increases the number of directory entries by 50 percent and dynamically allocates memory for the expanded array. (Use the Statistics (-y) parameter to check the current Directory Size (-D) value. If the amount of memory necessary for allocating the expanded array is exceeded or if the size of the expanded array exceeds 31K, Progress exits the session.) When the limit is reached, Progress issues a WARNING message, which is written to the current output (if there is current output) and to the log file (LG).

You can force Progress to adhere to the specified Directory Size (-D) limit by starting the session with the Hardlimit (-hardlimit) startup parameter. When you use the -hardlimit startup parameter, Progress issues a WARNING message when you exceed the Directory Size limit. It also issues a message that a resource limit was reached and raises an untrappable STOP condition. Note that -hardlimit also enforces the limits set by the Local Buffer Size (-l), the Maximum Memory (-mmax), and the Nested Blocks (-nb) startup parameters, so the STOP condition is raised when any of the specified limits are reached.

Date Format (-d)

Operating System and Syntax	UNIX Windows	-d *dateform*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	mdy	mdy

dateform

The format used to display dates in an application.

Use Date Format (-d) to specify the format used to display dates in an application using a three-character string of the letters: d, m, and y in any order. Note that -d sets the **display** format and not the **storage** format, which is fixed. Note that the date constants entered in procedures, or as initial values in the Data Dictionary, are always specified in month/day/year format.

DataService (-DataService)

Operating System and Syntax	UNIX Windows	-DataService *data-service*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC	–	–	–	–

data-service

The name of the DataService.

Use DataService (-DataService) to connect through a NameServer to an ODBC or ORACLE DataServer. The value for data-service must be a valid name from the DataService list registered to this NameServer, as defined by your appServiceName List entry in the broker properties file. If a default DataService has been defined for your broker instance, you can omit this parameter and connect using the default service.

Physical Database Name (-db)

Operating System and Syntax	UNIX Windows	-db *physical-dbname*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC	–	–	–	–

physical-dbname

The database to connect to when a Progress session is started. Specify the **physical** (not the **logical**) database name. If you are connecting in single-user mode or through shared memory, and the database is not in your current directory, you must specify the full path of the database.

Use Physical Database Name (-db) to connect one or more Progress databases. Use Single-user Mode (-1) after the name of the database to connect in single-user mode. You do not have to precede the first database on the command line with -db. However, for all subsequent databases that you want to connect, precede the database name with -db. For example, the following command connects two databases, abc and xyz:

```
pro abc -B 30 -db xyz -B 40
```

Debugger (-debug)

Operating System and Syntax	UNIX Windows	-debug		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

Use Debugger (-debug) to start a Progress session by running the Progress Debugger in stand-alone mode. This makes the Debugger (not the Procedure Editor) the point of entry for all application procedures to run. The Debugger begins by allowing breakpoints before using the Debugger RUN command to execute the startup procedure. Also, Progress automatically sets a breakpoint on the first executable line of the startup procedure before it runs.

For more information on the Debugger, see the *Progress Debugger Guide*.

Debug Alert (-debugalert)

Operating System and Syntax	UNIX Windows	-debugalert		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

You can use the -debugalert startup parameter along with the -clientlog startup parameter to access 4GL stack trace information during a session.

If you set Debug Alert to yes, and use the -clientlog startup parameter to specify a log file name, the 4GL stack trace is written to the log file you specify with -clientlog. Messages are written to the log file as they are encountered during a session. This applies to both batch and interactive sessions. For WebSpeed and AppServer applications, the output goes to their respective log files.

The stack trace is included only for the error messages and 4GL Alert-box messages that are normally written to the file specified with -clientlog.

NOTE: If you do not use the -clientlog startup parameter to specify a log file, the messages are not written to a log file.

When -debugalert is set to yes and an error occurs in a batch session for:

- A 4GL batch client, Progress records the 4GL stack trace information in the log file specified with -clientlog. In this case, Progress lists the stack trace information in reverse order. That is, the procedure that generated the error is written first, and the root procedure is written last.
- An AppServer client or a WebSpeed agent, Progress records the 4GL stack trace information in their respective log files, also in reverse order. If you view one of these log files in the Progress Explorer, it appears in top-down order since Explorer displays log files in reverse order.

To retrieve or reset the value of this parameter at runtime, use the DEBUG-ALERT attribute of the SESSION system handle.

For more information on the DEBUG-ALERT attribute or the SESSION system handle, see the *Progress Language Reference* and the *Progress Version 9 Product Update Bulletin*.

Dictionary Expressions (-dictexps)

Operating System and Syntax	UNIX Windows	-dictexps		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

Use Dictionary Expressions (-dictexps) to specify that Progress uses dictionary and help validation for all fields in all frames when you compile an application. Progress continues compiling even after it encounters a validation error.

This parameter serves as a temporary step to close possible validation holes in existing applications. It has the effect of adding a USE-DICT-EXPS option to every frame in the application, including those that are not used for input. This parameter closes all possible validation holes, but it is highly inefficient.

You can override this functionality for a particular frame by using the NO-VALIDATE keyword for that frame. See the Frame Phrase reference entry in the *Progress Language Reference* for more details on the NO-VALIDATE option.

If you do not specify -dictexps, Progress uses dictionary and help validation for a field only if that field is used in an UPDATE, SET, PROMPT-FOR, or ENABLE statement. In addition, Progress does not use dictionary and help validation when compiling if you specify only the *widget*:SENSITIVE=YES construct to make a widget sensitive. Progress stops compiling when it encounters a validation error.

Direct I/O (-directio)

Operating System and Syntax	UNIX NT	-directio		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC, DBS	–	–	Not enabled	Not enabled

Use Direct I/O (-directio) to open all files in unbuffered mode, which enables Progress to use an I/O technique that bypasses the operating system buffer pool and transfers data directly from a buffer to disk. This technique has several advantages over buffered reads and writes such as avoiding the overhead of maintaining the operating system buffer pool and eliminating competition for operating system buffers between Progress programs and other programs. The operating system buffer-pool algorithms are designed for efficient sequential file access; the Progress buffer-pool algorithms are more efficient for access to a Progress database.

You might improve Progress performance by using the direct I/O feature. To use direct I/O, use Blocks in Database Buffers (-B) to increase the size of the Progress buffer pool, since Progress I/O does not pass through the operating system buffer pool. Also, decrease the size of the operating system buffer pool to compensate for the additional memory allocated to Progress.

NOTE: Use asynchronous page writers (APWs). They improve database performance by performing overhead operations in the background.

DataServer (-Dsrv)

Operating System and Syntax	UNIX Windows	-Dsrv keyword (, value) [, keyword2 (, value2) ...]		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DS	–	–	–	–

value, value2...

A parameter for a DataServer.

Use DataServer (-Dsrv) to tell Progress that the specified keywords are parameters for the ODBC, ORACLE, or Progress/400 DataServer. These parameters are specific to ODBC data sources or are required to connect to certain data sources using the Progress DataServers. You can specify up to 50 keywords with the -Dsrv parameter.

There are three types of DataServer parameters:

- Progress DataServer for ODBC parameters
- ODBC SQL connection and statement parameters
- Progress DataServer for ORACLE parameters

For more information about the parameters, see the Progress DataServer guides.

Database Type (-dt)

Operating System and Syntax	UNIX Windows	-dt *db-type*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC, DS	–	–	Progress	Progress

db-type

The database type.

Use Database Type (-dt) to specify one of the following database types: Progress, ODBC, ORACLE, and AS/400. The value is not case sensitive. When connecting to a non-Progress database, use Database Type (-dt) to specify the type of database.

European Numeric Format (-E)

Operating System and Syntax	UNIX Windows	-E		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	AMERICAN	AMERICAN

Use European Numeric Format (-E) parameter to tell Progress to display thousands separators as periods and decimal points as commas (for example: 1.234.567,89) when displaying or prompting for numeric values.

AMERICAN is the default value, in which periods are displayed as decimal points and commas as thousands separators (for example: 1,234,567.89).

When you specify EUROPEAN, continue to use periods as decimal points and commas as thousands separators in your source files. Note that there is no command line parameter to specify AMERICAN, or reset EUROPEAN.

Numeric format specifications in the Data Dictionary and the decimal constants in procedures must always use the period to represent the decimal point.

You can also use the SESSION:NUMERIC-FORMAT to get or set the numeric format.

ESQL Log (-esqllog)

Operating System and Syntax	UNIX Windows	-esqllog		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
OID	–	–	–	–

Use ESQL Log (-esqllog) to tell Progress to send all messages to the database log file as well as to the message queue, which is available by the function call sqlgetmsg( ). See the *Progress Embedded SQL-89 Guide and Reference* for more information.

ESQL No Padding (-esqlnopad)

Operating System and Syntax	UNIX Windows	-esqlnopad		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
OID	–	–	–	–

Use ESQL No Padding (-esqlnopad) to enable the dynamic SQL scanner and the SQL C preprocessor to treat all hyphens as hyphens and let the application ensure that minus signs are recognized correctly. See the *Progress Embedded SQL-89 Guide and Reference* for more information.

Event Level (-evtlevel)

Operating System and Syntax	NT	-evtlevel *value*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	Normal	Normal

value

Use Event Level (-evtlevel) to specify the level of information that Progress writes to the NT Application Event Log. Valid values include:

- None: No Progress events are written to the NT Event Log.
- Brief: Progress Error and Warning messages are written to the NT Event Log.
- Normal: Progress Error and Warning messages are written to the NT Event Log along with any Progress message that is normally written to the log file (.lg). This is the default.
- Full: Progress Error, Warning, and Informational messages are written to the NT Event Log along with any messages generated by the Message Statement.

For more information about Progress and the NT Event Log, see the *Progress Installation and Configuration Guide Version 9 for UNIX* and *Progress Installation and Configuration Guide Version 9 for Windows*.

Expand Browse (-expandbrow)

Operating System and Syntax	Windows	-expandbrow		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–		

The -expandbrow startup parameter sets the EXPANDABLE and FIT-LAST-COLUMN option to on for all browsers created in the current session. The EXPANDABLE attribute has the same behavior as the FIT-LAST-COLUMN attribute. Therefore, if the -expandbrow parameter is specified at startup, the FIT-LAST-COLUMN attribute is set to TRUE for each browse in that session.

NOTE: Progress recommends that you use the FIT-LAST-COLUMN attribute instead of the EXPANDABLE attribute. This recommendation includes replacing EXPANDABLE with FIT-LAST-COLUMN in your current code.

For more information see *Progress Version 9 Product Update Bulletin* and the *Progress Language Reference*.

Force Access (-F)

Operating System and Syntax	UNIX Windows	-F		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC	–	–	–	–

Use Force Access (-F) with the PROSHUT Command to invoke an emergency shutdown of a shared-memory database.

NOTE: The PROMON and PROUTIL utilities also use a -F parameter. However, the results are not the same. In addition, using -F with PROUTIL may compromise the integrity of the database. For more information, see the *Progress Database Administration Guide and Reference*.

Schema Field Cache Size (-fc)

Operating System and Syntax	UNIX Windows	-fc *num-entries*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	0	128	128

num-entries

The number of entries in the schema field cache.

Use Schema Field Cache Size (-fc) to change the number of entries in the schema field cache. For example, you might increase the size if you retrieve large numbers of fields from your database. The Progress schema field cache reduces the time required to compile SQL queries by storing schema field information in memory. For best results, set this parameter to the total number of fields retrieved from the database.

Progress uses approximately 150 additional bytes of memory per schema field cache entry used. Because memory is allocated when required, unused entries produce minimal overhead.

If you set the value to 0, Progress disables schema field caching.

Field List Disable (-fldisable)

Operating System and Syntax	UNIX Windows	-fldisable		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

Use Field List Disable (-fldisable), which is a run-time parameter, to cause Progress to ignore field lists in the r-code and fetch complete records.

For more information about using field lists, see the *Progress Programming Handbook*.

Filtering Asynchronous COM Events (-filterocxevents)

Operating System and Syntax	UNIX Windows	-filterocxevents		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

The -filterocxevents startup parameter, lets you control the conditions under which you want asynchronous COM events handled. An asynchronous COM event is one that is sent by an ActiveX control or Automation object because of an external event, rather than in response to user input. One example of this is the Tick event from the PSTimer. Another is an event signalling the receipt of a message from an external source, such as a control that implements a mail notification system.

By default, asynchronous COM events are handled when any of the following conditions occur:

1. While an application is waiting for user input, such as during a WAIT–FOR or UPDATE statement.

2. While an alert box from a MESSAGE statement is on the screen.

3. During trigger processing, if the MultitaskingInterval parameter is set to a number greater than 0.

Handling COM events during the second and third conditions might cause random problems in your application, depending on what 4GL code is executed in the event trigger. You use the -filterocxevents startup parameter to prevent asynchronous COM events from being handled during the second and third conditions.

Before-image Cluster Age (-G)

Operating System and Syntax	UNIX Windows	-G *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	–	60	–	–

n

The number of seconds Progress waits.

Use Before-image Cluster Age (-G) to specify the number of seconds before Progress reuses a before-image cluster.

Group Delay (-groupdelay)

Operating System and Syntax	UNIX Windows	-groupdelay *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	1,000	0	0	1

n

The number of milliseconds a transaction waits before committing.

Use Group Delay (-groupdelay) to increase performance when Delayed BI File Write (-Mf) is set to zero. When the Group Delay is set greater than zero (0), Progress uses a technique known as Group Commit. When using Group Commit, a transaction spools its end note to the BI buffer and waits a short time until the buffer becomes full and is written to disk, or waits for other transactions to end and store their end notes in the BI buffer so that several transactions are committed by the same synchronous write. In this manner, Group Commit benefits overall performance, although each individual transaction might take slightly longer.

Host Name (-H)

Operating System and Syntax	Windows (SNA)	-H PLU=*plu-name*		
	UNIX Windows (TCP)	-H { *host-name* | localhost[1] }		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC, DBS, DS	–	–	–	–

[1] `localhost` does not apply to DataServers.

plu-name

On Windows systems running the Progress/400 DataServer, specifies the Partner LU name. The default for Windows 95 is the AS/400 host name from the CONFIG.PCS file.

host-name

The name (address) of the database server machine. This name is assigned to the machine in your TCP/IP hosts file.

`localhost`

A reserved word that specifies that the Database server communicate only with clients on the database server machine.

Use Host Name (-H) to identify the host name.

NOTE: This parameter has a special purpose when used with the Progress/400 DataServer and SNA. If you are connecting to multiple AS/400 machines, you must supply this parameter for each AS/400.

Number of Databases (-h)

Operating System and Syntax	UNIX Windows	-h *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC	240	1	5	5

n

The maximum number of databases that can be connected during a Progress session.

Use Number of Databases (-h) to connect to more than five databases during a single Progress session. Progress automatically allocates enough space to allow the connection of five databases.

Hardlimit (-hardlimit)

Operating System and Syntax	UNIX Windows	-hardlimit		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

Use Hardlimit (-hardlimit) to force Progress to adhere to the set limits for Directory Size (-D), Local Buffer Size (-l), Maximum Memory (-mmax), and Nested Blocks (-nb). These startup parameters are by default soft limits; that is, Progress tries to manage system resources within the limits given, but when it cannot, Progress increases the exceeded limit.

When -hardlimit is specified and one of the limits is exceeded, Progress generates a WARNING message that is written to the current output (if there is current output) and to the log file (LG). In addition, Progress issues a message that a resource limit was reached and raises an untrappable STOP condition.

Hash Table Entries (-hash)

Operating System and Syntax	UNIX Windows	-hash *n*	
Use With	**Maximum Value**	**Minimum Value**	**Default**
CC, DBS	–	13	Approximately 1/4 of the -B value

n

The number of hash table entries to use for the buffer pool.

CAUTION: Do not use this parameter unless directed to do so by Progress Software Technical Support.

No Crash Protection (-i)

Operating System and Syntax	UNIX Windows	-i		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC, DBS	–	–	–	–

Use No Crash Protection (-i) to tell Progress to run without using database integrity or database recovery. When running Progress without database integrity, it writes fewer data and before-image blocks to the disk. In this mode, some Progress procedures (such as those that create and delete large numbers of records) run significantly faster than if they are running with database integrity.

When running Progress with the -i parameter, transaction undo is supported. Therefore, there will still be a before-image file, which might grow quite large during very long transactions.

Use this parameter to do bulk data loading or for large batch runs. It reduces the number of disk input or output operations. Loading a database for the first time is a good example of a use for this parameter.

CAUTION: If you run Progress with the -i parameter and Progress fails for any reason, you cannot recover the database.

Do not use the -i parameter unless you have a complete backup of the database and can rerun procedures in case of a system failure. If the system fails during a Progress session started without crash protection, restore the backup copy and rerun the necessary procedures. For information on restoring a database, see the *Progress Database Administration Guide and Reference*.

The following messages might appear when starting a Progress session after a system failure:

- After a system failure while a full-integrity Progress session was running:

```
**The last session was abnormally terminated.
**Any incomplete transactions are being backed out.
**Database recovery is complete. You must rerun all active transactions.
```

- After a system failure while a no-integrity Progress session was running:

```
**The last session was abnormally terminated.
**The last session was run with the -i no integrity option.
**Your database cannot be repaired, you must restore a backup copy.
```

Dynamics Parameter (-icfparam)

Operating System and Syntax	UNIX Windows	-icfparam *string*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

string

A character string that supplies dynamics procedures with Dynamics-related data.

Use the Dynamics (-icfparam) parameter to specify a character string that can be accessed from 4GL procedures within the Dynamics framework. You can access this string within a 4GL procedure by reading the ICFPARAMETER attribute of the SESSION system handle.

NOTE: This parameter is reserved for use by Dynamics and procedures that have been integrated with Dynamics. Using this parameter for any purpose other than operating within the Dynamics framework will interfere with your ability to integrate your application with that framework at a later time.

For more information on the ICFPARAMETER attribute or the SESSION system handle, see the *Progress Language Reference*.

Index Range Size (-indexrangesize)

Operating System and Syntax	UNIX Windows	-indexrangesize *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	–	–	–	–

n

The number of indexes for which you want to track access statistics.

Use Index Range Size (-indexrangesize) to specify the number of indexes for which you want to collect statistics. For more information on tracking index statistics, see the *Progress Database Administration Guide and Reference*.

Initialization File (-ininame)

Operating System and Syntax	Windows	-ininame *name*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

name

Name of the initialization registry subkey or the application's initialization (.INI) file.

Use Initialization File (-ininame) to specify the location, in the registry, for the application's initialization information. When the Registry Base Key (-basekey) parameter is not specified at startup, Progress searches the HKEY_CURRENT_USER key followed by the HKEY_LOCAL_MACHINE key for the specified name. If no registry key of the specified name is found, Progress searches for a .INI file that matches *name*.

You can only use this startup parameter at the command line. It is ignored when used in a .pf file.

NOTE: The -ininame parameter replaces the -name option for the Windows Attributes (-Wa) parameter for Version 7.3B and later.

Input Characters (-inp)

Operating System and Syntax	UNIX Windows	-inp *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	32,000	1	4,096	4,096

n

The number of characters allowed in a single statement. The default is 4,096 characters.

Use Input Characters (-inp) to expand the available buffer space for a single statement. If not in use and 4,096 characters is exceeded in a statement, Progress displays a message stating that the statement is too long and advises you to increase the number of input characters using the -inp parameter.

AS/400 Ignore Stamp (-is)

Operating System and Syntax	UNIX Windows	-is		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC, DS	–	–	–	–

Use Ignore Stamp (-is) to stop a Progress client from comparing time stamps when attempting to connect an AS/400 database. When connecting to the schema holder database and the AS/400 database, Progress compares the time stamps on the files in the schema holder and the AS/400 physical and logical files. If the time stamps do not match, the login attempt is rejected.

NOTE: Whenever **any** change to the AS/400 database structure is made (whether or not the change is saved), the AS/400 data dictionary time stamp changes.

Progress checks the time stamp at the initial connection. If an AS/400 user modifies the file definitions after a successful Progress connect, the Progress user might get unpredictable results.

Initial Value Segment No Convert (-isnoconv)

Operating System and Syntax	UNIX Windows	-isnoconv		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

Use Initial Value Segment No Convert (-isnoconv) to disable a code page conversion that Version 9.1A provides and that previous versions do not. You might consider using -isnoconv when all three of the following conditions occur:

- A 4GL program contains character variables initialized from text literals.
- The text literals contain international characters.
- The code page of the r-code (-cprcodein) does not match the code page of internal memory (-cpinternal).

When you ran earlier versions of Progress under these conditions, when Progress read text literals from r-code into internal memory, it did not convert the code page. To compensate for this, programmers often pre-converted international characters in text literals to -cpinternal.

However, when you run Version 9.1A under these conditions, when Progress reads text literals from r-code into internal memory, it converts the code page from -cprcodein to -cpinternal. This code page conversion might cause the following problems:

- International characters pre-converted to -cpinternal are converted again to -cpinternal, garbling them.
- The text segment grows beyond its maximum size.

Keyword Forget List (-k)

Operating System and Syntax	UNIX Windows	-k *filename*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

filename

The name of a file that contains Progress keywords to disable. Each keyword must use a single line in the file. If Progress encounters a word that is not a Progress keyword, it returns a warning message but continues processing the file. Progress uses the PROPATH variable to locate *filename*. If it cannot find *filename*, it displays an error message and does not start Progress.

Use Keyword Forget List (-k) to disable Progress keywords. The keyword forget list is provided to ease migration from one version of Progress to the next. Do not use keywords in procedures as object names (tables, fields, frames, variables, streams, etc.). If necessary, disable new keywords (and the features they implement) until they can be removed from the application.

The directory where Progress is installed contains the following keyword files:

- newkywd file, which lists all keywords that are new in the most recent version of Progress
- newkw79 file, which you should use instead of newkywd, if you are converting from Version 7 to Version 9
- newkw69 file, which you should use instead of newkywd, if you converting from Version 6 to Version 9

Lock Table Entries (-L)

Operating System and Syntax	UNIX Windows	-L *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	System dependent[1]	32	–	8192

[1] Limited by available memory.

n

The number of entries in the record locking table. If you specify a value that is not a multiple of 32, Progress rounds the value you specify to the next highest multiple of 32.

Use Lock Table Entries (-L) to change the limits of the record locking table. Each record that is accessed by any user takes one entry. This is true whether the record is accessed with SHARE-LOCK or EXCLUSIVE-LOCK. Increase the size of the lock table if the following message appears:

```
SYSTEM ERROR: Record lock table too small. Increase -L parameter.
```

This message might also indicate that a particular procedure should be restructured into smaller transactions or should be run in single-user rather than multi-user mode. When lock table limits are exceeded, check to make sure transactions are not too large before increasing the lock table size.

If a user process tries to acquire a lock and the lock table overflows, the user's program is aborted, but the server continues to operate. Any partial transactions are undone.

Note that two record locks are acquired when records are accessed with the BREAK BY option (in DO, FOR EACH, or REPEAT statements).

Each lock table entry takes 18 bytes on typical systems, but 14 bytes on nonshared-memory systems.

Local Buffer Size (-l)

Operating System and Syntax	UNIX Windows	-l *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	4,000,000,000	1	200	200

n

The size of the local record buffer in 1KB units.

Use Local Buffer Size (-l) to change the size of the local record buffer in which Progress stores all variables, work files, and records that are in use at one time for a user. The maximum size of the local record buffer is 4GB for most systems. If you get one of the following error messages, you need to increase the size of this buffer:

```
SYSTEM ERROR: bfget: No space. Increase -l parameter.
SYSTEM ERROR: bfxpnd: No space. Increase -l parameter.
```

The Local Buffer Size (-l) limit is a soft limit; so if your application needs to exceed the limit, Progress automatically increases it and issues a WARNING message. The WARNING is written to the current output (if there is current output) and to the log file (LG).

You can force Progress to adhere to the specified -l limit by starting the session with the Hardlimit (-hardlimit) startup parameter. When you use the -hardlimit startup parameter, Progress, issues the WARNING message when you exceed the Local Buffer Size limit. It also issues a message that a resource limit was reached and raises an untrappable STOP condition. Note that -hardlimit also enforces the limits set by the Directory Size (-D), the Maximum Memory (-mmax), and the Nested Blocks (-nb) startup parameters, so the STOP condition is raised when any of the specified limits is reached.

Logical Database Name (-ld)

Operating System and Syntax	UNIX, Windows	-ld *logical-dbname*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC	–	–	–	–

logical-dbname

The logical database name.

Use Logical Database Name (-ld) to assign the logical name to a database. If you omit this parameter, Progress uses the physical database name (without the path or .db suffix).

The name must be a valid identifier, which means:

- It can be up to 32 characters long.
- It can be any combination of English letters and numbers, underscore (_), and dash (-) characters.
- The first character must be an uppercase or lowercase letter.

A logical database name cannot include Progress 4GL reserved words, a space, any accented letters, or the following special characters:

```
\ " ' * ; | ? [ ] ( ) ! { } < >
```

For more information on logical database names, see the *Progress Programming Handbook* and *Progress Database Administration Guide and Reference*.

Lock Timeout (-lkwtmo)

Operating System and Syntax	UNIX Windows	-lkwtmo *seconds*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	60	1,800	1,800

seconds

The wait time, in seconds.

Use Lock Timeout (-lkwtmo) to specify a different wait time (in seconds). A Progress process encountering a locked resource waits for a limited time before continuing execution. If the resource is still locked, the process times out and the wait is canceled. This feature is called "lock wait timeout."

The default wait time is 1800 seconds (30 minutes). The minimum value is 60 seconds. If a user specifies a value less than 60 seconds, Progress silently sets the value at 60 seconds.

NOTE: The wait time is accurate to within one minute of the value specified.

Lock timeout applies to interactive and batch clients.

Lock timeout affects WAIT-FOR record locks, schema locks, and transaction commit locks. Lock timeout does not affect internal locks used inside the database manager, such as buffer locks.

Language (-lng)

Operating System and Syntax	UNIX Windows	-lng *language-name*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	"?"	"?"

language-name

A character string that contains the name of the current language. This value must be surrounded by quotes.

Use Language (-lng) to specify the initial return value for the CURRENT-LANGUAGE function. This setting determines from which r-code segment Progress reads character-string constants.

The default value for the Language parameter is a character string that contains the question mark (?) character, which is not to be confused with the UNKNOWN value. This tells Progress to use the default language for the application.

For more information on the Language startup parameter, see the *Progress Internationalization Guide* and *Progress Language Reference*.

Auto Server (-m1)

Operating System and Syntax	UNIX Windows	-m1		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	–	–	–	–

Use Auto Server (-m1) to start an auto server. The Progress broker uses the auto server internally to start a remote user server. This is the default. You will never have to use this parameter directly.

Manual Server (-m2)

Operating System and Syntax	UNIX NT	-m2		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	–	–	–	–

Use Manual Server (-m2) to manually start a remote user server after you start a broker (servers are generally started automatically by the broker process). Use this parameter in the following cases:

- For debugging purposes, to start servers directly and observe their behavior
- On systems where automatic server generation is not possible

Secondary Login Broker (-m3)

Operating System and Syntax	UNIX NT	-m3		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	–	–	–	–

In a network environment where more than one broker is using the same protocol, use Secondary Login Broker (-m3) to start each secondary broker. The secondary broker logs in clients and starts remote user servers.

Maximum Clients Per Server (-Ma)

Operating System and Syntax	UNIX NT	-Ma *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	2048	1	–	5 users/server

n

The maximum number of remote users per database server. The default is the Maximum Number of Users (-n) parameter value, divided by the Maximum Number of Servers (-Mn) parameter value.

Use Maximum Clients per Server (-Ma) to specify the maximum number of remote users per database server. The Maximum Clients per Server (-Ma), Minimum Clients per Server (-Mi), and Maximum Servers (-Mn) startup parameters pertain only to versions of Progress that use shared memory. In addition, these parameters apply only to databases that are accessed from remote network nodes.

In most cases, the default behavior is desirable. Note that the default calculation is usually high because it assumes that all users are remote users, while the number specified with -n includes local users. If servers become overloaded with clients, reset the -Mn parameter to increase the number of servers.

NOTE: For SQL-92 database brokers, there is a limit of one client per server. The maximum value does not apply.

If experience proves that a given number of remote clients overloads a server or exhausts the file descriptors on the system, set -Ma to limit clients per server below that level. A *file descriptor* is an object that UNIX uses to identify a particular file. The file descriptors might be exhausted because Progress uses them to identify sockets.

Maximum Dynamic Server (-maxport)

Operating System and Syntax	UNIX NT	-maxport *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	System dependent	System dependent	–	65,535

n

The port number that is the highest in a specified range.

Use Maximum Dynamic Server (-maxport) to specify the highest port number in a specified range of port numbers accessible to a client. You specify the lowest port number with the -minport parameter. The range of port numbers defined by the -maxport and -minport parameters provides client access to a Progress server that is behind a firewall. Some operating systems choose transient client ports in the 32,768-to-65,535 range. Choosing a port in this range might produce unwanted results.

Delayed BI File Write (-Mf)

Operating System and Syntax	UNIX NT	-Mf *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	32,768	0	0[1]	3[2]

[1] Default is 3 for batch jobs.

[2] Default is 0 for nonshared-memory systems.

n

Any positive value delays Progress from synchronously writing out to disk the last before-image (BI) file records at the end of each transaction. On UNIX systems using shared memory, it also specifies the interval that the broker process wakes up to make sure all BI file changes have been written to disk. The default is 3 for single-user batch jobs and for multi-user databases using shared memory. Otherwise, the default is 0.

Use Delayed BI File Write (-Mf) to improve performance on a heavily loaded system. Using -Mf does not reduce database integrity. However, if there is a system failure, it is possible the last few completed transactions will be lost (never actually written to the BI file).

When running with full integrity, at the end of each transaction Progress does a synchronous write to disk of the last BI file block. This write guarantees that the completed transaction is recorded permanently in the database. If the user is notified that the transaction has completed and the system or database manager crashes shortly afterwards, the transaction is not lost.

Do not set -Mf on a lightly loaded system with little database update activity. Under these conditions, the extra BI write is very important and does not impact performance. On a heavily loaded system, however, the BI write is less important (the BI block will be written to disk very soon anyway), and has a significant performance penalty. Setting -Mf to delay this extra BI write saves one write operation per transaction, which can significantly improve performance. The extra BI file write is delayed by default for batch jobs.

If -Mf is set to a positive value, the last BI file record is only guaranteed to be written out to disk when a user logs out, or when the server or broker process terminates normally. On multi-user systems, the *n* argument determines the oldest completed transaction that can be lost.

Minimum Clients per Server (-Mi)

Operating System and Syntax	UNIX NT	-Mi *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	–	1	–	1

n

The number of remote users on a server before the broker starts another server. See also the Maximum Servers (-Mn) startup parameter.

Use Minimum Clients per Server (Mi) to specify the number of remote users on a server before the broker starts another server (up to the maximum number of servers). This parameter pertains only to multi-user versions of Progress that use shared memory. In addition, -Mi and -Mn apply only to databases that are accessed from remote network nodes.

As remote users enter the database, the broker process starts just one server process for each *n* remote users, until the maximum number of servers (specified by the -Mn parameter) is started. If you specify a value of 1, the broker starts a new server for each of the first -Mn remote users. Subsequent remote users are distributed evenly among the servers until the maximum number of users (-n) or maximum clients per server (-Ma) limits are reached.

NOTE: If you are using SQL-92, you do not need to use -Mi because the limit is always 1 client per server.

Typically, you can leave -Mi and -Mn at their default values. If you significantly increase -Mn, you should also increase -Mi. For example, if you set -Mn to 10 to accommodate up to 40 or more remote users, increase -Mi to 3 or 4 to prevent a situation where 10 servers were started for just 10 remote users.

Minimum Dynamic Server (-minport)

Operating System and Syntax	UNIX NT	-minport *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	–	1,025	–	1,025

n

The port number that is the lower in a specified range.

Use Minimum Dynamic Server (-minport) to specify the lowest port number in a specified range of port numbers accessible to a client. You specify the higher port number with the -maxport parameter. Ports below 1025 are usually reserved for system TCP and UDP. The range of port numbers defined by the -maxport and -minport parameters provides 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.

Message Buffer Size (-Mm)

Operating System and Syntax	UNIX Windows	-Mm *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	32,600	350	–	1,024

n

The message buffer size.

Use Message Buffer Size (-Mm) to specify the standard message buffer size, in bytes. This parameter is relevant only for network protocols; it is ignored for messages passed using UNIX pipes or NETBIOS.

Progress uses message buffers to move records (messages) between servers and remote clients. Records (plus 40-byte headers) larger than the message buffer size are fragmented into multiple messages. If your database records are large, increase this parameter to avoid record fragmentation. However, if the network works more efficiently with small messages, reduce -Mm and fragment larger records.

NOTE: On a given database, you must specify the same message buffers size when starting the server or broker (using PROSERVE) and multi-user clients (using MPRO) and when shutting down the database (using PROSHUT). That is, if you change the -Mm value on any database command, you must change all PROSERVE Command, MPRO Command, and PROSHUT Commands accordingly.

Maximum Memory (-mmax)

Operating System and Syntax	UNIX Windows	-mmax *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	65,534	1	512	512

n

The amount of memory allocated for r-code segments.

Use Maximum Memory (-mmax) to change the initial amount of memory allocated for r-code segments, in kilobytes. Progress dynamically allocates space for r-code segments in the execution buffer as needed. When memory allocation reaches the value specified by the Maximum Memory (-mmax) value, Progress writes nonactive segments to the sort file to make room for new active procedures. (Progress writes library-stored r-code to the sort file only if you specified the PROLIB Swap (-pls) startup parameter.) If you have large procedures or deeply nested procedure calls, you can use -mmax to increase the initial size of the execution buffer to reduce disk I/O activity required to swap segments to the sort file.

The -mmax value does not limit the amount of memory available for r-code segments. If Progress requires more space in the execution buffer and cannot regain enough space by swapping inactive segments to the sort file, Progress allocates more memory.

The -mmax limit is a soft limit; so if your application needs to exceed the limit, Progress automatically increases it and issues a WARNING message. The WARNING is written to the current output (if there is current output) and to the log file (LG).

You can force Progress to adhere to the specified -mmax limit by starting the session with the Hardlimit (-hardlimit) startup parameter. When you use the -hardlimit startup parameter, Progress issues the WARNING message when you exceed the Maximum Memory limit. It also issues a message that a resource limit was reached and raises an untrappable STOP condition. Note also that -hardlimit also enforces the limits set by the Directory Size (-D) the Local Buffer Size (-l), and the Nested Blocks (-nb) startup parameters. The STOP condition is raised when any of the specified limits is reached.

Use Statistics (-y) and Segment Statistics (-yd) to see segment allocation information.

Maximum Servers (-Mn)

Operating System and Syntax	UNIX NT	-Mn *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	512	1	–	4

n

The maximum number of remote client servers that can be started on the system.

Use Maximum Servers (-Mn) to limit the number of remote user servers that can be started by the broker process. The performance tradeoff to consider is swapping overhead for many servers versus overloading (slowing down) a server with too many clients. For more information on controlling memory use, see the *Progress Database Administration Guide and Reference*.

This parameter pertains only to multi-user versions of Progress that use shared memory. In addition, this parameter applies only to databases that are accessed from remote network nodes.

Also, use Minimum Clients per Server (-Mi) to adjust the actual number of servers in use. See the Maximum Clients Per Server (-Ma) and Minimum Clients per Server (-Mi) startup parameters for more information.

Servers Per Protocol (-Mp)

Operating System and Syntax	UNIX NT	-Mp *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	Value of -Mn	1	–	value of -Mn

n

The number of servers a broker can start.

Use Servers per Protocol (-Mp) with Secondary Login Broker (-m3) in database networks that use more than one network protocol. This parameter limits the number of servers that the broker can start to serve remote users for any one protocol. The total number of servers for all protocols is still limited by the Maximum Servers (-Mn)) parameter.

Maximum Servers Per Broker (-Mpb)

Operating System and Syntax	UNIX NT	-Mpb *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	–	–	–	–

n

The number of servers multiple brokers can start.

Use Maximum Server Per Broker (-Mpb) to specify the maximum number of servers that multiple brokers can start to serve remote users for any one protocol. For more information, see the *Progress Database Administration Guide and Reference*.

VLM Page Table Entry Optimization (-Mpte)

Operating System and Syntax	Digital UNIX	-Mpte		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	–	–	–	–

Use VLM Page Table Entry Optimization (-Mpte) to allocate shared memory in multiples of 8 MB at server startup for VLM64 support. This function is a binary switch that is off by default. The -Mpte startup parameter turns the function on.

Record Buffer Size (-Mr)

Operating System and Syntax	UNIX Windows	-Mr *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	32,000	96	–	1,012

n

The standard record buffer size in bytes.

On some systems, you can use Record Buffer Size (-Mr) to improve performance by varying the standard record buffer size. However, the default value of 1,024 bytes is appropriate in nearly all cases.

The record buffer size dictates that compaction or “garbage collection” is performed on the user's local buffer. If the largest contiguous fragment of free memory in the local buffer falls below the record buffer size in length, Progress automatically consolidates all free space.

Free-space consolidation also occurs when your process attempts to read in a database record that is larger than the largest block of free space available. Consolidation is more expensive in this case, because the record-copying process is interrupted for free-space consolidation when the record does not fit, and the copying process must then be restarted. To avoid record recopying, set -Mr larger than the database record size (1,024 is larger than the records in most databases). The tradeoff is more occurrences of (less expensive) automatic free space consolidation.

Shared-memory Overflow Size (-Mxs)

Operating System and Syntax	UNIX NT	-Mxs *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	Varies[1]	1	–	4KB + (*n* * 300)

[1] The maximum is limited only by the size of the signed integer data type on the system.

n

The size of the shared-memory overflow area in kilobytes.

Use Shared-memory Overflow (-Mxs) to replace the default value of the shared-memory overflow area; it does not increase it. The overflow area is appended to the shared-memory area. This parameter is relevant only on shared-memory systems.

If the overflow area is too small, Progress exits with the following message:

```
SYSTEM ERROR: Out of free shared memory. Use -Mxs to increase.
```

Depending on the operating system, Progress rounds the shared-memory area size to the next 512-byte or 4K boundary.

Network Type (-N)

Operating System and Syntax	UNIX Windows	-N *network-type*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC, DBS	–	–	–	System dependent

network-type

The network communications protocol.

Use Network Type (-N) to specify the network communications protocol supported by Progress: TCP or SNA (Progress/400). SNA allows Windows and certain UNIX clients to access Progress/400 servers. For more information, see the *Progress/400 Product Guide*.

Number of Users (-n)

Operating System and Syntax	UNIX Windows	-n *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	10,000	–	–	20

n

The maximum number of Progress users on the system. After *n* users have connected to the Progress database, additional user startup attempts are rejected.

On systems that do not use shared memory, use Number of Users (-n) to limit the total number of users below the level that overloads the database server.

On shared-memory systems, -n must be high enough to include local and remote users as well as background writers (APWs, BIWs, and AIWs), PROWDOG processes, and PROMON sessions. For more information, see the Minimum Clients per Server (-Mi) and Maximum Servers (-Mn) startup parameters.

Nested Blocks (-nb)

Operating System and Syntax	UNIX Windows	-nb *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	255	20	50	50

n

The maximum number of nested blocks.

Use Nested Blocks (-nb) to limit the maximum number of nested procedure blocks allowed. By default, the maximum number of nested blocks is 50. Nested block entries are allocated in memory; each requires 12 bytes. Therefore, decrease -nb only if memory is severely limited.

The -nb limit is a soft limit; so if your application needs to exceed the limit, Progress automatically increases it and issues a WARNING message. The WARNING is written to the current output (if there is current output) and to the log file (LG).

You can force Progress to adhere to the specified -nb limit by starting the session with the Hardlimit (-hardlimit) startup parameter. When you use the -hardlimit startup parameter, Progress, issues the WARNING message when you exceed the Nested Blocks (-nb) limit. It also issues a message that a resource limit was reached and raises an untrappable STOP condition. Note that -hardlimit also enforces the limits set by the Directory Size (-D), the Local Buffer Size (-l), and the Maximum Memory (-mmax) startup parameters, so the STOP condition is raised when any of the specified limits is reached.

No Lock (-NL)

Operating System and Syntax	UNIX Windows	-NL		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS, OID (Version 7.3B and later)	–	–	–	–

Use No Lock (-NL) to have all record retrieval statements default to NO-LOCK record access. In order to get the NO-LOCK default for any procedure, it must be **compiled** in a Progress session started with -NL. Running Progress with -NL has no affect on precompiled procedures.

When a transaction ends, EXCLUSIVE locks are downgraded to SHARE-LOCK, regardless of the -NL parameter.

For Version 7.3B and later, you can specify -NL as a startup parameter for the Open Interface Driver to support ODBC applications that perform multiple connections to the same data source, lock all records in a read table, or that otherwise exceed the lock table size with large databases.

No Check Temp-table Names (-nochkttnames)

Operating System and Syntax	UNIX Windows	-nochkttnames		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

Use the No Check Temp-Table Names (-nochkttnames) parameter to suppress the limited checking of TEMP-TABLE column names when passing TEMP-TABLE as parameters to procedures.

When passing TEMP-TABLE as a parameter to a procedure, the calling procedure's TEMP-TABLE and the called procedure's TEMP-TABLE must match with respect to the number of columns, the data type of each column, and the number of extents of each column (for columns with extents).

Progress provides limited checking of TEMP-TABLE column names when passing TEMP-TABLE as parameters to procedures. That is, for any given column:

- If the name of a column in the calling procedure's TEMP-TABLE does not match any column defined in the called procedure's TEMP-TABLE, Progress processes the column.
- If the name and position of a column in the calling procedure's TEMP-TABLE matches a column defined in the called procedure's TEMP-TABLE, Progress processes the column.
- If the name of a column in the calling procedure's TEMP-TABLE matches a column defined in the called procedure's TEMP-TABLE, but the positions do not match, Progress reports an error and the called procedure does not run.

Use the No Check Temp-table Names (-nochkttnames) parameter when you want Progress to process the TEMP-TABLE without verifying column names.

NOTE: TEMP-TABLE column names are case-insensitive.

Index Hint (-noindexhint)

Operating System and Syntax	UNIX Windows	-noindexhint		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DS	–	–	–	–

Use Index Hint (-noindexhint) to stop a DataServer for ORACLE from providing index hints to the ORACLE DBMS, which the DataServer does by default. Generally, index hints improve performance, but ORACLE's responses to hints varies among releases. This parameter is valid only when using the DataServer for ORACLE.

Server Join (-nojoinbysqldb)

Operating System and Syntax	UNIX Windows	-nojoinbysqldb		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DS	–	–	–	–

Use Server Join (-nojoinbysqldb) to specify that the client evaluate and performs queries that have joins. This might slow performance, but provides results that are consistent with Progress behavior. It also allows you to run DataServer applications on Version 9 clients with Version 8 servers. It overrides the DataServer default behavior, which is to instruct the non-Progress server to perform the join.

Use -nojoinbysqldb at compile time. It has no effect at run time.

SELECT Pass Through Disable (-noSQLbyserver)

Operating System and Syntax	UNIX Windows	-noSQLbyserver		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DS	–	–	–	–

Use SELECT Pass Through Disable (-noSQLbyserver) to turn off SELECT Pass Through for the current Progress session. Use with DataServers for ODBC and ORACLE.

Fractional Separator (-numdec)

Operating System and Syntax	UNIX Windows	-numdec *numeric-value*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	. (period)	. (period)

numeric-value

The numeric value of the character that represents the decimal point.

Use Fractional Separator (-numdec) to change the character that represents a number's decimal point in formatted text. Progress converts *numeric-value* to its character equivalent using the code page corresponding to the Internal Code Page (-cpinternal) startup parameter.

The decimal point cannot be represented by any of the following:

- The characters B C D R Z z 0 1 2 3 4 5 6 7 8 9 + - < > ( ) * ?
- The space character
- Any multi-byte character

As an example, to change the decimal point in formatted text to a comma (whose numeric value in the ISO8859-1 code page is 44), use:

```
-numdec 44
```

The default decimal point is the period (.).

Thousands Separator (-numsep)

Operating System and Syntax	UNIX Windows	-numsep *numeric-value*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	, (comma)	, (comma)

numeric-value

The numeric value of the character that represents the thousands separator.

Use Thousands Separator (-numsep) to change the character that represents a number's thousands separator in formatted text. Progress converts *numeric-value* to its character equivalent using the code page corresponding to the Internal Code Page (-cpinternal) startup parameter.

The thousands separator cannot be represented by any of the following:

- The characters B C D R Z z 0 1 2 3 4 5 6 7 8 9 + - < > ( ) * ?
- Any multi-byte character

NOTE: You can represent the thousands separator with the space character.

As an example, to change the thousands separator in formatted text to an apostrophe (whose numeric value in the ISO8859-1 code page is 39), use:

```
-numsep 39
```

The default thousands separator is the comma (,).

Printer (-o)

Operating System and Syntax	UNIX Windows	-o *printername*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	System dependent[1]	System dependent[1]

[1] See defaults listed below.

printername

Identifies the printer to use when processing the OUTPUT TO PRINTER statement in procedures.

On Windows:

Use Printer (-o) followed by the name of the printer port. For example:

```
-o LPT1
```

If you do not use the -o parameter, Progress uses the printer specified as the default printer.

On UNIX:

Use Printer (**-o**) followed by the name of the spooler and any necessary parameters. If you do not use the -o parameter, Progress defaults to “lp” (System V) or “lpr” (BSD). If you are passing arguments with this parameter and are using a script such as PRO or one of your own, enclose the entire string in quotes so that the argument to -o is returned as a single value when passed to the Progress module:

```
-o "lp -s"
```

Progress uses the spooler or printer port named at execution time rather than using the name at compile time. That way, precompiled procedures work regardless of which spooler or printer port is in effect.

To use a print spooler with spooler parameters, use OUTPUT TO PRINTER and specify the spooler parameters with -o. Use one of the following commands:

```
OUTPUT THROUGH lp spooler-parameters PAGED.
```

```
OUTPUT THROUGH lpr spooler-parameters PAGED.
```

Outer-join Mode (-ojmode)

Operating System and Syntax	UNIX Windows	-ojmode *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	2	1	1	1

n

An integer specifying the join mode.

Use the Outer-join Mode (-ojmode) parameter to specify the mode in which mixed inner and left outer joins, in queries of three or more joined tables, are processed.

Mixing inner and left outer joins lets you filter and reduce the amount of data you see on the left side of your left outer joins. When mixing these two types of join, inner joins take precedence over left outer joins. That is, the last inner join in a query causes the results of a prior outer join in the query to be like an inner join. This is because any rows that contain missing data from a prior left outer join are eliminated by the following inner join, if the data from that inner join is also missing. If the subsequent inner join has data, rows containing missing data from the prior left outer join are not eliminated.

If you set the value of -ojmode to 1, mixed inner and left outer joins in queries of three or more joined tables are processed as described above. Use this join mode when you want to see rows that contain missing data from a prior left outer join and existent data from a subsequent inner join. This is the default join mode.

If you set the value of -ojmode to 2, left outer joins take precedence over inner joins. That is, any rows that contain missing data from a prior left outer join are not eliminated by a subsequent inner join, whether the data from that inner join is missing or not. Use this join mode when you want to see rows that contain missing data from both a prior left outer join and a subsequent inner join.

NOTE: In any query, keeping your inner joins contiguous on the left with any left outer joins contiguous on the right will produce the same result in either join mode.

Password (-P)

Operating System and Syntax	UNIX Windows	-P *password*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC	–	–	–	–

password

The password of the user connecting to the database.

Use Password (-P) together with the User ID (-U) to specify the user password. Progress checks the _User table for the user ID supplied with the -U parameter. When it finds that user ID, it compares the password supplied with the -P parameter with the password in the _User table. If the two passwords match, Progress establishes the database connection.

With certain DataServers, the -P and -U parameters pass DataServer login information to the non-Progress database. For information, see your DataServer documentation.

Startup Procedure (-p)

Operating System and Syntax	UNIX Windows	-p *filename*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	_edit.p	_edit.p

filename

The name of the procedure to run when starting Progress.

Use Startup Procedure (-p) to specify a procedure that executes at the start of a Progress session. A common use for the -p parameter is to run a Progress procedure that displays a main application menu. The user can then choose options from the menu to run other Progress procedures. As a built-in security measure, when users press **CTRL-C** while running a procedure specified by -p, Progress prevents the users from accessing the procedure editor by rerunning the specified procedure (unless the application explicitly allows procedure editor access).

If you use the Batch (-b) startup parameter (MBPRO or BPRO), also use the -p parameter.

Parameter (-param)

Operating System and Syntax	UNIX Windows	-param *string*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

string

A character string that supplies information to a 4GL application. Can also be a comma-separated list of files to open in Procedure Editor buffers.

Use Parameter (-param) to specify a character string that can be accessed from 4GL procedures. You can also use this parameter to specify one or more files to load into Procedure Editor buffers. The Procedure Editor only loads the specified files when you use this parameter with a startup command that places you directly in the Procedure Editor. The files are not loaded when you use the parameter with a startup command that places you in the Progress Application Development Environment (ADE) desktop.

Within a 4GL procedure, you can access the string by reading the PARAMETER attribute of the SESSION system handle.

Parameter File (-pf)

Operating System and Syntax	UNIX Windows	-pf *filename*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC, CS, DBS, DS, OID	–	–	–	–

filename

The name of a parameter file that contains startup parameters to run Progress.

Use Parameter File (-pf) to name a parameter file that includes any number of startup parameters to run Progress. This parameter is especially useful if you regularly use the same parameters to start Progress, or if more parameters are specified than can fit on the command line. This parameter can be included within the parameter file itself to reference another parameter file.

Use multiple instances of -pf to name multiple parameter files. This allows you to specify application-specific parameters in one parameter file, database-specific parameters in a second parameter file, and user-specific parameters in yet another file.

See Chapter 1, “Introduction,” for more information about parameter files.

Pin Shared Memory (-pinshm)

Operating System and Syntax	UNIX Windows	-pinshm		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	–	–	–	–

The Pin Shared Memory (-pinshm) parameter does not have any arguments.

Using -pinshm will prevent the database engine from swapping shared memory contents to disk, which can help improve performance.

PROLIB Memory (-plm)

Operating System and Syntax	UNIX Windows	-plm		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

Use PROLIB Memory (`-plm`) to allocate a 512byte cache for a standard library directory. Otherwise, when using a standard library, the library's internal directory is loaded into memory. The directory is used to access the r-code files stored in the library. This parameter slows the library's speed of access but increases available memory. If you specify `-plm` with memory-mapped libraries, Progress ignores it. For more information on r-code libraries, see the *Progress Client Deployment Guide* and *Progress Programming Handbook*.

PROLIB Swap (-pls)

Operating System and Syntax	UNIX Windows	-pls		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

Use PROLIB Swap (`-pls`) to direct Progress to store the r-code files locally in temporary sort files.

When accessing r-code files stored in a standard library, Progress generally reads the files directly from the library instead of swapping the files into temporary sort files. However, over a network, direct reads might actually take longer than reading r-code files once and storing them locally in temporary sort files.

If you specify `-pls` with memory-mapped libraries, Progress ignores it. For more information on r-code libraries, see the *Progress Client Deployment Guide* and *Progress Programming Handbook*.

Fast Schema Change (-populate)

Operating System and Syntax	UNIX Windows	-populate		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC	–	–	–	–

Use Fast Schema Change (`-populate`) to turn off fast schema change when you are adding fields to a table. Fast schema change is the default behavior.

In earlier versions of Progress, adding a field to an existing database table is a time-consuming process for large databases. The DBMS updated the data in each row of the table to reflect the newly added field. Now Progress provides a fast schema change when you are adding fields to a table when all qualifying conditions are met. If all conditions are not met, adding a field causes Progress to update every row in the table.

ESQL PROPATH (-pp)

Operating System and Syntax	UNIX Windows	-pp *directory*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
OID	–	–	–	–

directory

The name of the new directory to add to the existing PROPATH. You can specify where the Open Interface Driver or ESQL small client searches for the r-code files.

If your applications require r-code files, use ESQL PROPATH (-pp) to specify the PROPATH for each small client's r-code files. Otherwise, the Open Interface Broker does not know where the r-code files for each small client reside. Before starting an ESQL small client, ensure that any precompiled r-code files reside in a directory accessible to the Open Interface Driver. The ESQL PROPATH (-pp) parameter is valid only with ESQL.

Configuration Properties File (-properties)

Operating System and Syntax	UNIX Windows	-properties *filename*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	–	–	–	–

filename

The name of the properties file.

Use Configuration Properties File (-properties) to identify the properties file that the AdminServer uses internally to specify startup parameters when starting a database server or servergroup. The default is $DLC/properties/conmgr.properties. You do not use this parameter directly.

Proxy Host (-proxyhost)

Operating System and Syntax	UNIX Windows	-proxyhost { *host-name* | *IP-address* }		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC	–	–	–	–

host-name

The name of the host at which the HTTP-based proxy server is located.

IP-address

The IP address of the host at which the HTTP-based proxy server is located.

Use the Proxy Host (`-proxyhost`) parameter to specify the name of the host or the IP address of the host at which the HTTP-based proxy server is located.

When you specify this parameter, all connections made by an AppServer client using the HTTP or HTTPS protocol connect to an AppServer Internet Adapter (AIA) instance using the proxy server at the specified host and port.

NOTE: If you specify the Proxy Host (`-proxyhost`) parameter, you must also specify the Proxy Port (`-proxyport`) parameter.

Proxy Password (-proxyPassword)

Operating System and Syntax	UNIX Windows	-proxyPassword <*password*>		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

password

Password used to authenticate the AppServer client to the HTTP-based Proxy server. *password* can be a string of up to 512 printable ASCII characters.

You should use -proxyPassword when -proxyhost and -proxyport are specified and the Proxy server requires authentication.

You can access -proxyPassword using the SESSION:PROXY-PASSWORD attribute.

If -proxyhost and -proxyport are not specified on the command line to the 4GL client, any value for -proxyPassword, whether specified with the startup parameter or with the SESSION attribute, is ignored.

password is validated during the AppServer CONNECT() method. If the *password* is invalid, the CONNECT() method fails and Progress issues an error message.

If -proxyUserid is not specified or SESSION:PROXY-USERID is not set, the CONNECT() method ignores any value for -proxyPassword.

Proxy Port (-proxyport)

Operating System and Syntax	UNIX Windows	-proxyport { *port-number* }		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC	65536	1	–	–

port-number

The port number of the HTTP-based proxy server.

Use the Proxy Port (-proxyport) parameter to specify the port on which the HTTP-based Proxy server is listening.

When you specify this parameter, all connections made by an AppServer client using the HTTP or HTTPS protocol connect to an AppServer Internet Adapter (AIA) instance using the proxy server at the specified host and port.

If you specify the Proxy Port (-proxyport) parameter, you must also specify the Proxy Host (-proxyhost) parameter.

Proxy Userid (-proxyUserid)

Operating System and Syntax	UNIX Windows	-proxyUserid *<user-id>*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

user-id

> *user-id* used to authenticate an AppServer client to the HTTP-based proxy server. The *user-id* can be a string of up to 512 printable ASCII characters, including the space character.

You can access -proxyUserid using the SESSION:PROXY-USERID attribute.

If -proxyhost and -proxyport are not specified on the command-line to the 4GL client, any value for the -proxyUserid, whether specified with the startup parameter or with the SESSION attribute, is ignored.

The *user-id* is validated during the AppServer CONNECT() method. If *user-id* is invalid, the CONNECT() method fails and Progress issues an error message.

If -proxyPassword is not specified or SESSION:PROXY-PASSWORD is not set, and -proxyUserid is specified, a blank proxy password is used by the AppServer CONNECT() method.

ANSI SQL (-Q)

Operating System and Syntax	UNIX Windows	-Q		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
OID	–	–	–	–

Use ANSI SQL (-Q) as necessary to enforce strict ANSI SQL conformance when processing Progress/SQL statements. Progress/SQL meets or extends the ANSI SQL standard. However, there are a few cases where Progress extensions conflict with the ANSI SQL standard. Use -Q if you want strict adherence to the standard.

By default, Progress/SQL character value comparisons are not case sensitive. This conflicts with the ANSI standard. Progress character fields (columns) and variables are defined as **case insensitive** by default. If you use -Q, then for any CREATE TABLE statements executed during that session, all columns default to case sensitive. (Columns defined as case insensitive prior to running a Progress session are not affected by -Q.)

For more information on case sensitivity, see the DEFINE PARAMETER, DEFINE VARIABLE, and DEFINE WORKFILE reference entries in the *Progress Language Reference*.

Note that case sensitivity for columns and variables applies only to the stored data values; the actual table names, column names, and variable names are never case sensitive, nor are comparisons involving these names. For example, the following comparison is never case sensitive:

```
. . . WHERE _Field-name BEGINS 'ny'
```

The -Q parameter also enforces code that ensures that there are no temporary unique constraint violations.

ANSI SQL Client (-Q2)

Operating System and Syntax	UNIX Windows	-Q2		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
OID	–	–	–	–

Use ANSI SQL Client (-Q2) to set Progress so that clients conform to the rules governing the ANSI SQL89 guaranteed ability to serialize transactions. Only use -Q2 to conform to these standards. You can only use the -Q2 parameter with Progress ESQL.

Quick Request (-q)

Operating System and Syntax	UNIX Windows	-q		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

Use Quick Request (-q) to tell Progress to search PROPATH directories only on the first use of a procedure.

Ordinarily in a Progress procedure, when the RUN statement is used to run a subprocedure, Progress searches the directories named by the PROPATH environment variable, looking for a procedure of the same name with a .r extension. If it finds a file with a .r extension (an r-code file), it checks to make sure the r-code file has not changed since that r-code file was created.

This search is very useful in a development environment where procedures change regularly and you want to make sure you are always running the most current version of your application. However, in a production environment, you might want to bypass this search.

With Quick Request (-q), after the initial search, if the procedure still resides in memory or in the local session-compiled file, Progress uses that version of the procedure rather than searching the directories again. However, Progress always checks whether Data Dictionary definitions related to a procedure were modified. If they were modified, Progress displays an error when it tries to retrieve the procedure.

Non-reliable I/O (-r)

Operating System and Syntax	UNIX Windows	-r		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC	–	–	Unbuffered I/O	Unbuffered I/O

Use Buffered I/O (-r) to enable buffered I/O to the before-image file. In most cases, avoid using this parameter because it might put database integrity at risk.

CAUTION: If you run Progress with the -r parameter and your system fails because of a system crash or power failure, you cannot recover the database. You must restore the database from a backup and restart processing from the beginning.

CAUTION: If you are updating a database using buffered I/O and there is a Progress system failure but no hardware failure, Progress can recover the database.

In general, you always want complete database integrity or you want fast performance for large tasks, such as loading large amounts of data or doing fiscal year-end processing. The -r parameter does not guarantee complete database integrity, and No Crash Protection (-i) is faster in terms of performance.

Alternate Random Number Generator (-rand)

Operating System and Syntax	UNIX Windows	-rand *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	2	1	1	1

n

The type of random number generator. A value of 1 (the default) indicates that Progress should use the original generator; specify a value of 2 to use an alternate.

Use Alternate Random Number Generator (-rand) to specify whether to use the old random number generator or the new one. The old number generator always generates the same random sequence; that is, the numbers that it generates are random, but each time a session starts, it gives you the same set of numbers from the last session. If you have to generate a different sequence of random numbers, specify the new generator.

Reread Nolock (-rereadnolock)

Operating System and Syntax	UNIX Windows	-rereadnolock		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

Use Reread Nolock (-rereadnolock) to tell Progress that when it attempts to find a record with NO-LOCK, to re-read the record from the database, even if the record is already in a buffer. You can use this parameter to resolve client-server currency conflicts. You can also use it to resolve server-to-server currency conflicts by using it as an AppServer startup parameter via the Progress Explorer or by setting the srvrStartupParam property in the ubroker.properties file for the appropriate AppServer.

Keep the following in mind when you use -rereadnolock:

- It has no affect on records that are retrieved via RECID or ROWID. In that case, Progress will not re-read the record. Instead, it uses the copy of the record already stored in the buffer. If you need the most current version of the record, use the RELEASE statement on all buffers that contain a copy of the record before reading the record, or use the FIND CURRENT or GET CURRENT statement to re-read the record.
- It has no affect on the behavior of the query cache used for a query with NO-LOCK that is specified via the CACHE phrase of the DEFINE QUERY statement. To force Progress to always re-read the record, set the cache size to zero (0). However, this may significantly degrade performance if the database is accessed across a network. Set the cache size to zero only when it is critical to retrieve the most current version of a record.

- It has no affect on the behavior of the prefetch cache that is used by default when retrieving records NO-LOCK across the network. By default, when executing a CAN-FIND function or a FIND, FOR, or OPEN QUERY statement on a database that is accessed across a network, Progress fetches several records at a time and stores them in a prefetch cache. Progress will only send a request to the database server to fetch more records if the requested record is not in the current prefetch cache. If the record is in the current cache, Progress will not read a new copy of that record even if -rereadnolock is set. To eliminate this cache so that the most current version of the record is always read, use the NO-PREFETCH keyword in the appropriate statements. However, using the NO-PREFETCH keyword may significantly degrade performance. Only set NO-PREFETCH if it is critical to retrieve the most current version of a record.

Run 4GL Client (-rg)

Operating System and Syntax	UNIX Windows	-rg		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

Use Run 4GL Client (-rg) to indicate that you want to run the 4GL client. Use this parameter if you have more than one type of Progress client and want to limit access/functionally of the session to the 4GL client.

Read-only Media (-RO)

Operating System and Syntax	UNIX Windows	-RO		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC	–	–	–	–

Use Read-only (-RO) to open a database for read-only access. Use -RO to access a database stored on read-only media (a read-only optical disk, for example).

NOTE: Do not run servers for a read-only database. Progress does not support servers for read-only users.

If you use the -RO parameter when other users are updating the database, you might see invalid data, such as stale data or index entries pointing to records that have been deleted.

A read-only session is essentially a single-user session. Read-only users do not share database resources (database buffers, lock table, index cursors). However, unlike a normal single-user database, a read-only database can be simultaneously accessed by multiple -RO users.

When a read-only session starts, it does not check for the existence of a lock file for the database. Furthermore, a read-only user opens the database file, but not the log or before-image files. Therefore, read-only user activity does not appear in the log file.

If a database is opened with -RO, the string returned by the DBRESTRICTIONS function includes the keyword READ-ONLY.

License Usage Reporting Interval (-rptint)

Operating System and Syntax	UNIX NT	-rptint *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	–	–	–	–

n

The interval, in minutes, at which the server polls for usage information.

Use License Usage Reporting Interval (-rptint) to define the polling interval at which the server gathers license usage statistics reported in the usage report file. If you omit this parameter, no usage reporting occurs. The -rptint parameter is relevant only on shared-memory systems.

NOTE: The recommended minimum value is 30. If you set this value below 30, the high polling frequency will cause your log file to grow unreasonably large; it might also severely impact server performance.

You must shut down the server to change this parameter.

If you specify the -rptint parameter, Progress saves the following information in a text file named *dbname*.rpt:

- Time stamp of the entry
- Current user count
- Maximum number of concurrent users since the last entry
- Maximum number of concurrent users since server startup
- Total number of users since server startup

- Number of attempts to exceed user-defined user count (defined with the User Count (`-usrcount`) parameter) since server startup
- Number of attempts to exceed the licensed user count since server startup

For more information about license usage reporting, see the *Progress Installation and Configuration Guide Version 9 for UNIX* and *Progress Installation and Configuration Guide Version 9 for Windows*.

Run Query Client (-rq)

Operating System and Syntax	UNIX Windows	-rq		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

Use Run Query Client (-rq) to indicate that you want to run the Query client. Use this parameter if you have more than one type of Progress client and want to limit access or functionality of the session to the Query client.

Run Run-time Client (-rr)

Operating System and Syntax	UNIX Windows	-rr		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

Use Run Run-time Client (-rr) to run the Run-time Client. Use this parameter if you have more than one type of Progress client and want to limit access or functionality of the session to the Run-time Client.

Encrypted Compiler Mode (-rx)

Operating System and Syntax	UNIX Windows	-rx		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

Use Encrypted Compiler Mode (-rx) to enable query or run-time Progress clients to compile encrypted source code and access the Data Dictionary to manage schema (for example: for security management and to dump or load .df files). If Progress is started with -rx, you can only compile encrypted source code. Use -rx when you are shipping an application in the form of encrypted source code to end-user sites having query or run-time versions of Progress. You must prepare a tool that end users can run to compile your code, and this tool must include a command that invokes Progress with -rx.

Service Name (-S)

Operating System and Syntax	UNIX Windows	-S { *service-name* / *port-number* }		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC, DBS	–	–	–	–

service-name

The name of the service to be used by the broker process.

port-number

The port number of the host; if using Progress Explorer, the port number of the NameServer.

Use Service Name (-S) to specify the service or port number to be used when connecting to a broker process or used by a broker process on the host machine. You must use this parameter when you are starting:

- A broker or server on a machine that will serve remote users
- Multi-user Progress as a remote user

The system administrator must make an entry in the services file that specifies the server or broker name and port number.

When the broker spawns a server, the server inherits all of the network parameters (except the Service Name parameter) from the broker. Because there is no restriction on the number of brokers you can start, you can have multiple brokers running with different network parameters. See also the Server Group (-servergroup) startup parameter description.

Table 4–1 shows how the broker, server, and remote client interpret each of their parameters when you use the -S parameter.

Table 4–1: Parameter Interpretation With Service Name (-S)

Module	Interpretation
Broker	Parameters apply to the connections on which the broker is listening for connection requests from remote clients.
Server	Parameters apply to the connection between the server and the remote client.
Remote Client	Parameters identify the connection parameters to the broker or the server.

To run multi-user Progress from a remote network node, use both the Host Name (-H) and Service Name (-S) parameters.

Stack Size (-s)

Operating System and Syntax	UNIX Windows	-s *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	5,242,878[1]	9	40	40

[1] Actual maximum value is limited only by available memory.

n

The size of the stack in 1KB units.

Use Stack Size (-s) to change the size of the stack (an internal memory area used by Progress program modules). Increase the stack size if one of the following messages appears:

```
SYSTEM ERROR: stkpush: stack overflow. Increase -s parameter.
SYSTEM ERROR: stkditem: stack overflow. Increase -s parameter.
```

Stack overflow errors are most likely when data definitions are loaded for very large tables or use recursive procedures.

Semaphore Sets (-semsets)

Operating System and Syntax	UNIX	-semsets *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	Maximum Number of Users +1	1	–	1

n

An integer specifying the number of semaphore sets available to the Progress broker.

Use Semaphore Sets (`-semsets`) to change the number of semaphore sets available to the Progress broker. When more than 1,000 users connect to a single database, there might be high contention for the semaphore set. If there is a lot of semaphore contention on a system, using multiple semaphore sets helps alleviate this contention and improve performance on high user counts. For more information on using semaphore sets, see the *Progress Database Administration Guide and Reference*.

Server Group (-servergroup)

Operating System and Syntax	UNIX Windows	-servergroup *name*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	–	–	–	–

name

Specifies the name of the logical collection of server processes.

Use Server Group (-servergroup) to identify the logical collection of server processes to start. The name you specify must match the name of a servergroup in the conmgr.properties file. Use the Progress Explorer tool to create servergroups and save them in the conmgr.properties file. Do not edit the conmgr.properties file directly. To start a database configuration use Progress Explorer or the DBMAN utility.

Type of Server to Start (-ServerType)

Operating System and Syntax	UNIX Windows	-ServerType [4GL | SQL | Both]		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	–	–	Both	Both

4GL

Indicates that the broker will only support 4GL servers.

SQL

Indicates that the broker will only support SQL-92 servers.

Both

Indicates that the broker supports both 4GL and SQL-92 servers.

Use the Type of Server to Start (-ServerType) parameter to limit the type of server the broker can start.

To change the value of -ServerType, use the config.properites file or the command line.

AS/400 Server Program Name (-Sn)

Operating System and Syntax	UNIX Windows (SNA Only)	-Sn		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC	–	–	–	–

Use Server Program Name (-Sn) with SNA to specify that the Server program start on the AS/400 with a Progress connection request. For more information, see the *Progress/400 Product Guide*.

Spin Lock Retries (-spin)

Operating System and Syntax	UNIX NT	-spin *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	–	0	–	0

n

The number of times a process tries to acquire a latch before pausing.

Use Spin Lock Retries (-spin) to set a value to use the spin lock mechanism or a system of semaphores and queues. If the value of *n* is greater than 0, then a spin lock algorithm is used for shared-memory data structures. When a process has to lock a shared-memory structure, the process tries up to *n* times to acquire the latch for that structure. If it has not acquired the latch in *n* tries, then the process pauses, or naps. The length of the pause increases gradually if the process repeatedly fails to acquire a latch. After the allotted nap time, the process wakes up and tries to acquire the lock again. If it fails to acquire the lock, it will retry up to the number of tries specified by *n*.

If the value of *n* is 0, a system of semaphores and queues is used instead of spin locks. The spin lock algorithm is much more efficient than using semaphores when you have multiple processors.

In some cases, if the value of *n* is 1, it **might** improve performance even on a single-processor machine because Progress uses the spin lock mechanism. Do not set this parameter to a value larger than 1 if your machine has only one processor.

On multi-processor machines, try a value of 2,000. If this causes too much CPU usage, reduce the value. If you have many fast processors, a value as high as 10,000 might be effective.

You can evaluate the `-spin` values through the Progress Monitor (PROMON utility) R&D options. See the *Progress Database Administration Guide and Reference* for more information on the PROMON utility. The -spin parameter controls the performance indicator called resource waits. By setting the `-spin` value higher, you can reduce the resource waits. Note that when setting the -spin value higher ceases to reduce the resource waits, continuing to set it higher can adversely effect CPU utilization. To view the resource waits value:

- Access PROMON and enter R&D at the main menu.
- Choose option 3. Other Displays, then choose option 1. Performance Indicators to view the resource waits. Resource waits is the last item reported in the listing.

SQL-92 Stack Size (-SQLStackSize)

Operating System and Syntax	UNIX Windows	-SQLStackSize *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	System Dependent	1	1	1

n

The size of the stack in 1MB units.

Use SQL-92 Stack Size (`-SQLStackSize`) to change the size of the SQL stack (an internal memory area used by SQL program modules).

Use the command line to change the value of -SQLStackSize.

Statement Cache Size (-SQLStmtCacheSize)

Operating System and Syntax	UNIX Windows	-SQLStmtCacheSize *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC	–	–	–	100

n

The number of statements to be stored in the SQL Statement Cache.

Use Statement Cache Size (-SQLStmtCacheSize) to set the number of statements that can be stored in the SQL cache.

Use the command line to change the value of -SQLStmtCacheSize.

Stash Area (-stsh)

Operating System and Syntax	UNIX Windows	-stsh *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	31	1	2	2

n

The number of 1KB blocks for the stash area.

Use Stash Area (-stsh) to specify the number of 1 KB blocks to allocate to the stash area, which Progress uses as temporary storage for modified index fields. The default size is sufficient for most index fields, but large character fields with a word index might require more space.

You can calculate an appropriate stash size by taking the size in bytes of the largest field with a word index, doubling that value, and adding 1KB. For example, if the largest word indexed field is 2,000 characters long, set -stsh to 5 to create a stash area of 5KB. If you do not allocate enough space, Progress terminates with a fatal error. If this occurs, restart Progress with a larger stash area.

Open Interface Driver Connection (-SV)

Operating System and Syntax	UNIX Windows	-SV *parameter-list*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
OID	–	–	–	–

parameter-list

A list of parameters that specify the connection between a Progress ESQL-89 small client or ODBC Driver and the Open Interface Broker (OIB)/Open Interface Driver (OID).

Use Open Interface Driver Connection (-SV) on an OIB/OID and ESQL-89 small client command line to identify the connection between each ESQL-89 small client and its OIB/OID. The parameter list directly follows -SV on the command line up to the next Physical Database Name (-db)parameter or end-of-line character. For the ODBC Driver, you specify the *parameter-list* without the -SV in the appropriate dialog box of the ODBC Administrator utility in Windows.

Only the ESQL-89 small client and ODBC Driver can specify database connection parameters. Any Physical Database Name (-db) parameter lists that the small client specifies can appear before or after the -SV parameter list on the command line. The OIB passes the parameters that affect each OID (including database connection parameters) to the OID.

You can only include one -SV parameter list on each OIB and ESQL-89 small client command line. If Parameter File (-pf) is included in the -SV parameter list, all parameters in the file, up to the first -db parameter, are part of the -SV parameter list. However, all parameters after and including the first -db in the parameter file are ignored, and are not passed to OIDs. Thus, the only parameters you can effectively put in this parameter file are for the OID connection.

Table 4–2 lists the parameters you can specify in this list for UNIX and Windows.

Table 4–2: Parameters Used With the -SV Parameter Progress

Parameter	Syntax
Host Name of the Open Interface Broker	-H
Auto Server[1]	-m1
Manual Server[2] (specified for a user-started server only)	-m2
Message Buffer Size	-Mm
Network Type	-N
Nested Blocks (Windows small client only)	-Nb
No Lock (Version 7.3B and later)	-NL
Parameter File[1]	-pf
ESQL PROPATH[1]	-pp
Service Name	-S

1 No need to specify-the OIB automatically provides these.

2 Required if you start an OID directly. For more information, see the *Progress Embedded SQL-89 Guide and Reference.*

For more information on specifying OID connection parameters for an ESQL-89 small client, see the *Progress Embedded SQL-89 Guide and Reference*.

Temporary Directory (-T)

Operating System and Syntax	UNIX Windows	-T *dirname*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	Working directory	Working directory

dirname

The directory in which to create temporary files.

Use Temporary Directory (-T) to specify a directory for temporary files. If you do not use this parameter, Progress puts these temporary files in the current working directory. On UNIX and Windows, these filenames begin with a prefix of lbi, dbi, pge, srt, or trp, and end with the process ID of the user's Progress session.

NOTE: Access the -T parameter using the SESSION:TEMP-DIR attribute. This attribute is read-only.

The following table shows descriptions of these prefixes:

Temp File	Usage
LBI	Local before-image (subtransaction undo)
DBI	Store temporary tables
PGE	Edit buffer contents
SRT	Temporary sort space; session compile storage
TRP	Store Data Dictionary changes until they are saved

On UNIX, Progress names these files uniquely for each user to avoid filename conflicts. Furthermore, Progress stores these files as “unlinked” with no visible name in the UNIX file system (unless you use Save Temp Files (-t).

On Windows, if two or more users share the same working directory and there is a conflict of temporary files, the following message appears:

```
Unable to open or create filename, error 3
```

Save Temp Files (-t)

Operating System and Syntax	UNIX	-t		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

Use Save Temp Files (`-t`) to make temporary files visible on UNIX. Typically, temporary files (LBI*number*, SRT*number*, PGE*number*, DBI*number*, and TRP*number*) are invisible (unlinked on UNIX) when Progress is running. In these filenames, *number* is the process ID of the user's Progress session. The `-t` parameter also makes the entries visible after Progress aborts. However, if Progress aborts, delete the temporary files yourself.

With or without the `-t` parameter, Progress always deletes the temporary files when it terminates normally.

Table Range Size (-tablerangesize)

Operating System and Syntax	UNIX Windows	-tablerangesize *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	–	–	–	–

n

The number of tables for which you want to track access statistics.

Use Table Range Size (`-tablerangesize`) to specify the number of tables for which you want to collect statistics. For more information on tracking table statistics, see the *Progress Database Administration Guide and Reference*.

Speed Sort (-TB)

Operating System and Syntax	UNIX Windows	-TB *blocksize*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	31	1	2	2

blocksize

The block size in kilobytes, to allocate when sorting records for reports and when rebuilding indexes. The default block size is 2K but can be reset anywhere from 1KB to 31KB.

Use Speed Sort (-TB) primarily to improve sort performance, particularly during index rebuild operations. If you increase the block size beyond 2KB, Progress sorts records faster but uses more memory and disk space in the process. If system memory is severely limited, you might actually choose to lower the block size.

Sort space is allocated in the SRT file, a temporary session file having a system-generated unique name with the prefix srt. Progress uses the SRT file to store session-compile modules and as temporary work space during sorting (including rebuilding indexes). Use -TB to set the SRT file block size-the increments by which the SRT file grows.

In summary, increase -TB to at least 8 to improve index rebuild operations by 15 percent or more. Or, decrease -TB to 1 to relieve severe Windows memory shortages during sort operations. See also the description of the Merge Number (-TM) startup parameter.

NOTE: Increasing the -TB value to 4 or more can improve network performance when Progress reads large r-code files across a network.

Merge Number (-TM)

Operating System and Syntax	UNIX Windows	-TM *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	32	1	5	5

n

The number of blocks or streams to be simultaneously merged during the sort process.

Use Merge Number (-TM) to increase the speed of the merge phase of the sort process (at the cost of increased temporary memory usage). See also the description of the Speed Sort (-TB) startup parameter.

Token (-tok)

Operating System and Syntax	UNIX Windows	-tok *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	System dependent[1]	1	1024	1024

[1] Limited by available memory.

n

The number of tokens allowed in a 4GL statement.

Use Token (-tok) to specify the maximum number of tokens allowed in a single 4GL statement (default 1,024 tokens). If you exceed the default number, Progress prompts you to use this startup parameter to increase the number of tokens allowed.

Trigger Location (-trig)

Operating System and Syntax	UNIX Windows	-trig { *dir-name* | *lib-name* }		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC	–	–	–	–

dir-name, *lib-name*

The location of the database 4GL triggers.

Use Trigger Location (-trig) to identify the name of the directory or library that contains the database 4GL triggers for a database. Library names must have a .pl extension. Progress interprets all other extensions as directory names. If you do not use Trigger Location (-trig) when connecting to the database, Progress searches the PROPATH for the triggers. For more information about database triggers, see the *Progress Programming Handbook*.

NOTE: If you are generating the local binary schema cache, do not connect to the database using Trigger Location (-trig) and the Schema Cache File (-cache) together.

User ID (-U)

Operating System and Syntax	UNIX Windows	-U *userid*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CC	–	–	–	–

userid

The user ID of the user.

Use User ID (-U) with Password (-P) to specify the user ID. Progress checks the _User table for the user ID supplied with the -U parameter. When it finds that user ID, it compares the password (after it has encoded the password) supplied with the -P parameter with the password in the _User table. If the two passwords match, Progress assigns the user ID to the Progress session. Otherwise, the user is not allowed to connect to the database.

With certain DataServers, the -U and -P parameters pass DataServer login information to a non-Progress database. For more information, see your DataServer documentation.

Use OS Locale (-useOsLocale)

Operating System and Syntax	Windows	-useOsLocale		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

The Use OS Locale (-useOsLocale) parameter sets certain default localization options in Progress clients. It causes Progress to set the SESSION system handle attributes shown in Table 4–3 based on the current Windows locale settings. The table also shows the startup parameters that individually affect the value of each attribute.

Table 4–3: Attributes Set By the Use OS Locale (–useOsLocale) Parameter

SESSION System Handle Attribute	Associated Startup Parameters
DATE-FORMAT	Date Format (-d)
NUMERIC-DECIMAL-POINT	Fractional Separator (-numdec)
NUMERIC-FORMAT	European Numeric Format (-E) Fractional Separator (-numdec) Thousands Separator (-numsep)
NUMERIC-SEPARATOR	Thousands Separator (-numsep)

If any of the startup parameters shown in the table appear on the same command line with the -useOsLocale parameter, any that follow -useOsLocale override the setting from -useOsLocale and any that -useOsLocale follows are overridden by the setting from -useOsLocale.

For more information on these SESSION system handle attributes, see the *Progress Language Reference*

User Count (-usrcount)

Operating System and Syntax	UNIX Windows	-usrcount *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
DBS	–	–	–	–

n

The number of users allowed to connect to a database before license usage reporting begins.

Use User Count (`-usrcount`) to change the threshold at which license usage reporting begins. License usage reporting lets you track the number of connections to a database.

By default, Progress reports license usage information in the log (`.lg`) file when the number of batch and interactive connections to a database reaches 85 percent of the licensed user count defined in the Progress configuration file. If you specify `-usrcount`, when the number of users connected to the database meets or exceeds the value specified for *number*, Progress writes the following information to the log file:

- Time stamp
- User ID of the user attempting to connect
- The number of attempts to exceed the number specified for `-usrcount` since the server was started
- The number of attempts to exceed the licensed user count since the server was started

This information appears in a nonfatal message similar to following license usage reporting messages sample.

```
09:03:00 Usr n: User defined threshold  threshold exceeded by userid on
device-name.
09:03:00 Usr n: n attempts to exceed the licensed user count n since
server startup.
09:03:00 Usr n: n attempts to exceed the user threshold n since server
startup.
```

Progress also writes license usage summary information to the log file when the server is started and shut down.

No usage reporting occurs if you purchased an unlimited license or if you specify the `-usrcount` parameter with a value of 0.

NOTE: You must shut down the server to change the user count specified by `-usrcount`.

On shared-memory systems, you can control the frequency at which the server polls for license usage information with License Usage Reporting Interval (-rptint). For more information, see the `-rptint` startup parameter. For more information about Progress user licensing, see the *Progress Installation and Configuration Guide Version 9 for UNIX* and *Progress Installation and Configuration Guide Version 9 for Windows*.

Version 6 Colon (-v6colon)

Operating System and Syntax	UNIX Windows	-v6colon		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

Use Version 6 Colon (-v6colon) to direct Progress to perform colon alignment for unlabeled fields, as in Version 6. If you do not specify this parameter, Progress moves the field two characters to the right.

Version 6 Query (-v6q)

Operating System and Syntax	UNIX Windows	-v6q		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

Use Version 6 Query (-v6q) to direct the Progress server to use only a single index when resolving FOR EACH statements. If you do not use this parameter, the server might use more than one index. Using more than one index is more efficient, but might change the order in which records are returned.

The -v6q parameter also causes cursor repositioning with FOR EACH statements, as in Version 6.

Windows Passthrough Printing (-Wa -wpp)

Operating System and Syntax	Windows	-Wa -wpp		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

Use Windows Passthrough Printing (`-Wa -wpp`) to enable WPP mode for the session. You can only use this startup parameter at the command line. It is ignored when used in a `.pf` file. It must be the last parameter in the command.

Windows Single Session (-wss)

Operating System and Syntax	Windows	-wss		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

Use Windows Single Session (-wss) to limit a user to running only one Progress client session at a time. This parameter applies only to a Windows GUI client. If an existing Progress session is running on the machine when -wss is specified, the new session terminates and the windows of the existing session are brought forward to alert the user. To avoid unnecessary processing of other parameters, use the -wss parameter first on the command line.

If more than one Progress application is installed on a user's machine, do not use the -wss parameter else it will prevent a user from running two or more applications at the same time. In addition, the -wss parameter does not recognize versions of Progress prior to Version 9.0.

Windows Exit - No Dialog (-wy)

Operating System and Syntax	Windows	-wy		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

Use Windows Exit - No dialog (-wy) to prevent Progress from prompting you for confirmation when Windows is shut down with Progress active.

Statistics (-y)

Operating System and Syntax	UNIX Windows	-y		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

Use Statistics (-y) to collect procedure access and usage statistics throughout the Progress session. At session end, Progress writes these statistics to the default output file client.mon. For detailed information on the output written to the client.mon file, see the *Progress Database Administration Guide and Reference*.

At startup, the -y parameter sends a report of all the startup parameters to the output file. This report includes all default values, overridden values, and values you set at startup. Unlike the other statistics that the -y parameter collects, this report is not written at session end and is not affected by the SHOW-STATS statement.

Note that the specific statistics displayed might change periodically as new Progress features are implemented.

The edit buffer map statistics are written each time a user exits from Progress or uses the SHOW-STATS statement. These lines list the procedures currently in the edit (or execution) buffer and their r-code sizes.

NOTE: If you cannot execute SHOW-STATS from the Procedure Editor or cannot add the statement to your Progress code (for example, if while using Run-time Progress), you can specify Statistics with CTRL-C (-yc) instead of the -y parameter. Both parameters behave the same way, except that -yc lets you use CTRL-C as a substitute for the SHOW-STATS statement.

The program-access statistics are written to the output file when the session ends or when you use the SHOW-STATS statement. In the output, temp file reads and writes are reads and writes to the SRT file, which stores each user's session compiles and active r-code files. The Bytes column is a cumulative total. The Stat file checks are recorded because they represent a relatively time-consuming system call. Unless you invoke Quick Request (-q), Progress makes a stat call each time a precompiled subprocedure is called with the RUN statement.

Progress places the default output file (`client.mon`) in the current working directory. However, you can specify a different output file by using the CLIENTMON environment variable. Simply set CLIENTMON to point to the file you want to use.

For example, in a UNIX environment, if you wanted to use a file named `stats` in the `/usr/tmp` directory, enter the following command at the system prompt:

```
CLIENTMON=/usr/tmp/stats; export CLIENTMON
```

Statistics with CTRL-C (-yc)

Operating System and Syntax	UNIX Windows	-yc		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

Use Statistics with CTRL-C (-yc) to allow you to press CTRL-C as a substitute for executing the SHOW-STATS statement. Use this parameter if you cannot execute SHOW-STATS from the Procedure Editor or cannot add the statement to your Progress code (for example, if while using Run-time Progress).

When you specify this parameter, it disables CTRL-C as an interrupt signal. In addition, pressing CTRL-C only emulates SHOW-STATS without the CLEAR option.

Segment Statistics (-yd)

Operating System and Syntax	UNIX Windows	-yd		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

Use Segment Statistics (`-yd`) to write segment statistics to the client monitor file (`client.mon`, by default). This provides:

- A breakdown of an r-code file by segment type, including the number of segments and their total size
- Read and write access to the sort file by segment type and access type, including the number of times the sort file was accessed and the number of bytes read from/written to the sort file

Four Digit Year Default (-yr4def)

Operating System and Syntax	UNIX Windows	-yr4def		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

Use Four Digit Year Default (-yr4def) to output a four-digit year from the EXPORT, MESSAGE and PUT UNFORMATTED statements that may use a two-digit year. This option causes all dates to use a four-digit year, regardless of what is specified by the Century year Offset (-yy) startup parameter.

If you do not use this parameter, the EXPORT, MESSAGE and PUT UNFORMATTED statements produce a two-digit year for the date if it is within the 100 year period defined by the Century Year Offset (-yy) startup parameter. If the date is outside of that 100 year period, these statements produce a four-digit year for the date.

Using -yr4def does not affect statements such as DISPLAY, UPDATE and PUT (without UNFORMATTED), nor any default schema or program variable date formats, which continue to use the default format 99/99/99. These statements have the FORMAT phrase where an overriding date format can be given, if the 4-digit year is desired, or the fields and variables can be defined with the explicit 99/99/9999.

Statistics With Cross-reference (-yx)

Operating System and Syntax	UNIX Windows	-yx		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	–	–

Use Statistics with Cross-reference (-yx) to collect procedure call statistics and write them to an output file. With this parameter, you can monitor information to answer the questions:

- How many calls were made in a given period of time?
- How long did a procedure spend executing?
- How often was a procedure swapped to and from the temporary file (SRT file)?

Progress places the default output file for the -yx parameter, proc.mon , in your current working directory. However, you can specify a different output file by using the PROCMON environment variable. Simply set PROCMON to point to the file you want to use.

For example, in a UNIX environment, if you want to use a file named stats in the /usr/tmp directory, enter the following command on the command line:

```
PROCMON=/usr/tmp/stats; export PROCMON
```

Use the SHOW-STATS statement to write the procedure call statistics to the output file. The CLEAR option sets all counters and timers (such as Calls and Time) to 0. When you exit Progress, the -yx parameter writes the procedure call statistics to the output file whether or not you use SHOW-STATS.

Figure 4–1 shows sample procedure call statistics as they appear in the output file.

```
              Wed Feb 6 16:19:05 2002
Procedure call statistics:      16:19:21
 Caller     Callee     Load Size    Calls   Rd Bytes     Reread   Time
 <top>      _edit       3789         1        3789         0        16
 _edit      _adeload    9542         1        9542         0        10
 _edit      _prostar    6931         1        6931         0        23
 _prostar   _login      5087         1        5087         0        73
 _edit      _setcurs     562         2         562         0        20
 _edit      _proedit  181029         1      181029         0     12180
 _proedit   _toollic    2570         1        2570         0       268
 _proedit   _trrpfile   1510         2        1510         0       140
 _proedit   _kvlist     2624         2        2624         0        26
 _proedit   _setcurs     562         4           0         0        63
 _proedit   _adehelp    3279         1        3279         0        97
```

Figure 4–1: Sample Output From -yx Startup Parameter

The fields that appear in the output file `proc.mon` are as follows:

Caller

The names of any calling procedures. The word <top> indicates that there was no calling procedure, and that you ran the called procedure directly from the Progress Procedure Editor.

Callee

The names of the called procedures.

Load Size

The size, in bytes, of each called procedure as Progress loads it into memory.

If you see a load size of 0, the called procedure is an uncompiled source procedure (`.p` file), or an r-code file (`.r` file) that was previously loaded into memory.

Calls

The number of times the Caller procedure calls the Callee procedure.

Rd Bytes

Generally, the called procedure's load size. However, if the procedure is swapped out of memory and later restored, Rd bytes equals the procedure's Load Size added to the number of bytes read from the SRT file. Rd Bytes grows larger each time the procedure is swapped out of memory and restored from the SRT file.

Reread

The number of bytes Progress reads from the SRT file to restore a Caller procedure that was overwritten. Progress restores only what is necessary to continue executing the Caller procedure. In some cases, this is less than the amount swapped to the SRT file.

Time

The total execution time of the called procedure, in milliseconds.

Century Year Offset (-yy)

Operating System and Syntax	UNIX Windows	-yy *n*		
Use With	**Maximum Value**	**Minimum Value**	**Single-user Default**	**Multi-user Default**
CS	–	–	1950	1950

n

A four-digit year (1990, for example). The default is 1950.

Use Century Year Offset (-yy) to determine the start of the 100-year period in which a date with a two-digit year is valid.

Some Progress applications reserve only two-digits for the year in the date format. When, for example, -yy is set at 1950, Progress determines if the two-digit year value is greater or less than 50. If the year is greater than 50, Progress assumes that the date is in the twentieth century. If the year is less than 50, Progress assumes that the date is in the twenty-first century.

Table 4–4 shows examples.

Table 4–4: Century Setting and Date Format

-yy	Year (As Supplied in DATE Format)	Result of Year Function
1900	50-99	1950-1999
	00-49	1900-1949
1950	50-99	1950-1999
	00-49	2000-2049
1980	80-99	1980-1999
	00-79	2000-2079

Notice that all two-digit year values expand into the 100-year period beginning with -yy.

To test the effect of -yy, start Progress with a different -yy value and run the following procedure:

```
DEFINE VARIABLE ddate AS DATE.
DISPLAY "Enter date or Press F4 to end." WITH FRAME msg.
REPEAT:
  SET ddate LABEL "Date" WITH SIDE-LABELS.
  DISPLAY YEAR(ddate) FORMAT "9999" LABEL "=> Year"
  WITH SIDE-LABELS.
END.
```

NOTE: If you use a hard-coded date containing a two-digit year in a .p file, Progress honors the -yy parameter and expands the two-digit year to a four-digit year during compilation. However, this may not match the runtime -yy. For this reason, Progress Software recommends that you use four digit years for hard-coded dates in programs. For example:

```
DEFINE VARIABLE hdate AS DATE INITIAL 01/01/2001.
```

NOTE: When you dump or load any database, the -yy setting you load with must match the -yy setting that was used for dumping, unless you use the Four Digit Year Default (-yr4def) startup parameter for dumping.

Index

Numbers

-1 startup parameter 4–2

4GL parameters
  client 3–6

A

ADE Desktop
  startup commands
    Windows 2–5

ADE r-code location (-baseADE) startup parameter 4–9

-adminport startup parameter 4–3

AdminServer Port (-adminport) startup parameter 4–3

After-image buffers 2–11

After-Image Buffers (-aibufs) startup parameter 4–4

After-image Stall (-aistall) startup parameter 4–5

After-image writer
  PROAIW utility 2–11
  stopping 2–11

-aibufs startup parameter 4–4

-aistall startup parameter 4–5

AIW, see also After-image writer

Alternate Random Number Generator (-rand) startup parameter 4–126

ANSI SQL (-Q) startup parameter 4–122

ANSI SQL Client (-Q2) startup parameter 4–123

ANSI SQL standard 4–122

APW, see also Asynchronous page writer

AS/400 Host Name (-H) startup parameter 4–61

AS/400 Ignore Stamp (-is) startup parameter 4–71

AS/400 Server Program Name (-Sn) startup parameter 4–142

Async Queue Size (-asyncqueuesize) startup parameter 4–6

Asynchronous page writer 2–12
  stopping 2–12

-asyncqueuesize startup parameter 4–6

Auto Server (-m1) startup parameter 4–79

Auto Server server type 4–79

B

-B startup parameter 4–7

-b startup parameter 4–8
  PROSHUT command 2–20

Base Index (-baseindex) startup parameter 4–11

-baseADE startup parameter 4–9

Basekey (-basekey) startup parameter 4–12

Base Table (-basetable) startup parameter 4–13

Batch (-b) startup parameter 4–8

Batch startup commands
  BPRO 2–6
  MBPRO 2–8
  single-user 2–3
  Windows 2–5

Before-Image Buffers (-bibufs) startup parameter 4–14

Before-Image Cluster Age (-G) startup parameter 4–59

Before-imaging 4–14

Before-Image Buffers (-bibufs) startup parameter 2–13

Before-image writer 2–13
  starting 4–14
  stopping 2–13

Before-imaging
  -Mf startup parameter 4–84

-bibufs startup parameter 2–13, 4–14

-bistall startup parameter 4–15

-bithold startup parameter 4–16

BIW, see also Before-image writer

Bleeding Record Lock (-brl) startup parameter 4–19

Block sizes
  sorting 4–154

Blocks in Database Buffers (-B) startup parameter 4–7

-bn startup parameter 2–20

Bold typeface
  as typographical convention 1–xii

-Bp startup parameter 4–17

-Bpmax startup parameter 4–18

BPRO command 2–6

-brl startup parameter 4–19

Brokers
  shutting down
    permissions 2–22

-Bt startup parameter 4–20

Buffer Size for Temporary Tables (-Bt) startup parameter 4–20

Buffered I/O (-r) startup parameter 4–125

Buffers
  after-image
    specifying 4–4
  before-image
    specifying 4–14
  blocks in database buffers 4–7
  changing size of local record 4–75
  edit 4–167
  local 4–93
  message 4–87
  private read-only 4–17
  server startup parameters 3–16

-by startup parameter 2–20

C

-C disconnect specified user
PROSHUT Command 2–20

-C list users
PROSHUT command 2–20

-c startup parameter 4–21

-cache startup parameter 4–22

Case Table (-cpcase) startup parameter 4–30

Century (-yy) startup parameter 4–175

Check Double-byte Enabled (-checkdbe) startup parameter 4–24

-checkdbe startup parameter 4–24

Client Connection startup parameters 3–12

Client Internationalization startup parameters 3–8

Client Logging (-clientlog) startup parameter 4–26

Client Performance startup parameters 3–3

Client Session startup parameters 3–2
4GL 3–6
performance 3–3
product-type 3–5

Client startup parameters
database connection 3–12
session statistics 3–10
windows session 3–11

-clientlog startup parameter 4–26

Clients
defined 1–2

Code page
establishing case rules 4–30
for printer output 4–34
for reading r-code text segments 4–35
for writing r-code text segments 4–36
identifying 4–32
identifying collation table 4–31
used to write log messages 4–33

Collation Table (-cpcoll) startup parameter 4–31

Colon alignment
-v6colon startup parameter 4–162

Commas
Progress interpretation 4–50

Communications Parameter File (-cp) startup parameter 4–29

Compiling
encrypted source code 4–135

Configuration Properties File (-properties) startup parameter 4–117

Connecting
over a wide area network 4–22
to non-Progress databases 4–49

Conversion Map (-convmap) startup parameter 4–25, 4–28

-convmap startup parameter 4–25, 4–28

-cp startup parameter 4–29

-cpcase startup parameter 4–30

-cpcoll startup parameter 4–31

-cpinternal startup parameter 4–32

-cplog startup parameter 4–33

-cpprint startup parameter 4–34

-cprcodein startup parameter 4–35

-cprcodeout startup parameter 4–36

-cpstream startup parameter 4–37

-cpterm startup parameter 4–38

-cs startup parameter 4–39

CTRL-C statistics 4–169

Cursor Size (-cs) startup parameter 4–39

D

-D startup parameter 4–40

-d startup parameter 4–41

Data
  bulk loading 4–65

Database
  connecting to 4–43
    over a wide area network 4–22

Database connection
  client startup parameters 3–12

Database integrity 4–65

Database names
  logical 4–76
  physical 4–43

Database quiet points 2–16

Database Server parameters 3–15

Database shutdown
  avoiding 4–5, 4–15, 4–16

Database Type (-dt) startup parameter 4–49

DataServer (-Dsrv) startup parameter 4–48

DataServer startup parameters 3–22

DataServers
  connecting through a NameServer 4–42
  Join by SQLDB 4–101
  ODBC 4–42, 4–48, 4–102
  ORACLE 4–42, 4–48, 4–101, 4–102
  ORACLE index hints 4–100
  Progress/400 4–48, 4–142
  specifying keywords 4–48
  starting a broker 2–14
  startup parameters 3–22
  using Server Join (-nojoinbysqldb) 4–101

DataService (-DataService) startup parameter 4–42

-DataService startup parameter 4–42

Date format (-d) startup parameter 4–41

-db startup parameter 3–12, 4–43

Debug Alert (-debugalert) startup parameter 4–45

-debug startup parameter 4–44

-debugalert startup parameter 4–45

Debugger (-debug) startup parameter 4–44

Decimals
  Progress interpretation 4–50

Delayed BI File Write (-Mf) startup parameter 4–84

-dictexps startup parameter 4–46

Dictionary Expressions (-dictexps) startup parameter 4–46

Direct I/O (-directio) startup parameter 4–47

-directio startup parameter 4–47

Directories
  for temporary files 4–150

Directory entries
  compiled procedures 4–40

Directory Size (-D) startup parameter 4–40

Disconnect specified user
PROSHUT command 2–20

Double-byte characters
check 4–24

-Dsrv startup parameter 4–48

-dt startup parameter 4–49

Dynamics Parameter (-icfparam) startup parameter 4–67

E

-E startup parameter 4–50

Edit buffers 4–167

Editing parameter files 1–5

Encrypted Compiler Mode (-rx) startup parameter 4–135

Encrypted source code 4–135

Error messages 4–66
displaying descriptions 1–xvii

ESQL
-SV startup parameter 4–148

ESQL Log (-esqllog) startup parameter 4–51

ESQL No Padding (-esqlnopad) startup parameter 4–52

ESQL PROPATH (-pp) startup parameter 4–116

ESQL-89 client parameters, see also Open Interface Driver

-esqllog startup parameter 4–51

-esqlnopad startup parameter 4–52

European Numeric Format (-E) startup parameter 4–50

Event-Level (-evtlevel) startup parameter 4–53

-evtlevel startup parameter 4–53

Expand Browse (-expandbrow) startup parameter 4–54

-expandbrow startup parameter 4–54

F

-F startup parameter 4–55
PROSHUT command 2–20

Fast Schema Change (-populate) startup parameter 4–115

-fc startup parameter 4–56

File descriptors 4–82

Files
client.mon 4–167
keyword 4–73
parameters 1–3
saving temporary 4–152
searching 4–124
startup.pf 1–3

Filtering Asynchronous COM Events (-filterocxevents) startup parameter 4–58

-filterocxevents startup parameter 4–58

Force Access (-F) startup parameter 4–55
PROSHUT command 2–20

Four Digit Year Default (-yr4def) startup parameter 4–171

Four-digit year 4–175

Fractional Separator (-numdec) startup parameter 4–103

Free space
consolidation of 4–93

G

-G startup parameter 4–59

Group Delay (-groupdelay) startup parameter 4–60

-groupdelay startup parameter 4–60

-Gw startup parameter 2–21

H

-H startup parameter 2–21, 4–61

-h startup parameter 4–62

Hardlimit (-hardlimit) startup parameter 4–63

-hardlimit startup parameter 4–63

-hash startup parameter 4–64

Hash Table Entries (-hash) startup parameter 4–64

Help
Progress messages 1–xvii

Help button
adding to alert boxes 4–45

Host Name (-H) startup parameter 4–61
PROSHUT command 2–21

I

-i startup parameter 4–65

-icfparam Dynamics startup parameter 4–67

Index Base (-baseindex) startup parameter 4–11

Index Cursors (-c) startup parameter 4–21

Index Hint (-noindexhint) startup parameter 4–100

Index levels, see also Cursor Size (-cs) startup parameter

Index Range Size (-indexrangesize) startup parameter 4–68

-indexrangesize startup parameter 4–68

-ininame startup parameter 4–69

Initial Value Segment No Convert (-isnoconv) startup parameter 4–72

Initialization File (-ininame) startup parameter 4–69

-inp startup parameter 4–70

Input Characters (-inp) startup parameter 4–70

Integrity
database 4–65, 4–125

Internal Code Page (-cpinternal) startup parameter 4–32

-is startup parameter 4–71

-isnoconv startup parameter 4–72

Italic typeface
as typographical convention 1–xii

J

Java classpath
identifying 4–25

K

-k startup parameter 4–73

Keystrokes 1–xiii

Keyword Forget List (-k) startup parameter 4–73

Kill users (-by) startup parameter
PROSHUT command 2–20

L

-L startup parameter 4–74

-l startup parameter 4–75

Language (-lng) startup parameter 4–78

Large transactions
effect on lock table 4–74

-ld startup parameter 4–76

License usage reporting 4–160

License Usage Reporting Interval (-rptint) startup parameter 4–131

List users
PROSHUT command 2–20

-lkwtmo startup parameter 4–77

-lng startup parameter 4–78

Local Buffer Size (-l) startup parameter 4–75

Lock Table Entries (-L) startup parameter 4–74

Lock Timeout (-lkwtmo) startup parameter 4–77

Locking
-NL startup parameter 4–98

Log File Code Page (-cplog) startup parameter 4–33

Log files
setting maximum size 4–16

Logical Database Name (-ld) startup parameter 4–76

M

-m1 startup parameter 4–79

-m2 startup parameter 4–80

-m3 startup parameter 4–81

-Ma startup parameter 4–82

Manual
syntax notation 1–xiii

Manual Server (-m2) startup parameter 4–80

Manual Server server type 4–80

Maximum Clients per Server (-Ma) startup parameter 4–82

Maximum Dynamic Server Port (-maxport) startup parameter 4–83

Maximum Memory (-mmax) startup parameter 4–88

Maximum Private Buffers per Connection (-Bpmax) startup parameter 4–18

Maximum Servers (-Mn) startup parameter 4–89

Maximum Servers per Broker (-Mpb) startup parameter 4–91

-maxport startup parameter 4–83

MBPRO command 2–8

Memory
  server startup parameters 3–16
  shared 4–94
  startup parameter 4–88

Merge Number (-TM) startup parameter 4–155

Message Buffer Size (-Mm) startup parameter 4–87

Messages
  displaying descriptions 1–xvii
  system failure 4–66

-Mf startup parameter 4–84

-Mi startup parameter 4–85

Minimum Clients per Server (-Mi) startup parameter 4–85

Minimum Dynamic Server (-minport) startup parameter 4–86

-minport startup parameter 4–86

-Mm startup parameter 4–87

-mmax startup parameter 4–88

-Mn startup parameter 4–89

Monospaced typeface
  as typographical convention 1–xii

-Mp startup parameter 4–90

-Mpb startup parameter 4–91

MPRO command 2–9

-Mr startup parameter 4–93

-Mxs startup parameter 4–94

N

-N startup parameter 2–21, 4–95

-n startup parameter 4–96

-nb startup parameter 4–97

Nested Blocks (-nb) startup parameter 4–97

Network Type (-N) startup parameter 4–95
  PROSHUT command 2–21

Networks
  connecting over wide area 4–22
  server startup parameters 3–20

newkywd file 4–73

-NL startup parameter 4–98

No Check Temp-table Names (-nochkttnames) startup parameter 4–99

No Crash Protection (-i) startup parameter 4–65

No kill users (-bn) startup parameter
  PROSHUT command 2–20

No Lock (-NL) startup parameter 4–98

-nochkttnames startup parameter 4–99

-noindexhint startup parameter 4–100

-nojoinbysqldb startup parameter 4–101

-noSQLbyserver startup parameter 4–102

NT Event Log
  specifying level of information 4–53

Number of Databases (-h) startup parameter 4–62

Number of Users (-n) startup parameter 4–96

-numdec startup parameter 4–103

Numeric format
changing 4–50, 4–103, 4–104

-numsep startup parameter 4–104

O

-o startup parameter 4–105

-ojmode startup parameter 4–107

Open Interface Driver
startup parameters 3–23

Open Interface Driver Connection (-SV) startup parameter 4–148

Outer-join Mode (-ojmode) startup parameter 4–107

OUTPUT TO PRINTER statement
startup parameter 4–105

P

-P startup parameter 4–108

-p startup parameter 2–6, 2–8, 4–109

Page writer, see also Asynchronous page writer

-param startup parameter 4–110

Parameter (-param) startup parameter 4–110

Parameter File (-pf) startup parameter 1–4, 4–111

Parameter files 1–3
creating 1–5
editing 1–5
format 1–6
invoking 1–4
specifying 1–4

Password (-P) startup parameter 4–108

Performance
server startup parameters 3–16

-pf startup parameter 4–111

Physical Database Name (-db) startup parameter 4–43
location 3–12

-pinshm startup parameter 4–112

-plm startup parameter 4–113

-pls startup parameter 4–114

Polling interval
startup parameter 4–131

-populate startup parameter 4–115

-pp startup parameter 4–116

-properties startup parameter 4–117

Print Code Page (-cpprint) startup parameter 4–34

Printer (-o) startup parameter 4–105

Private Read-only Buffers (-Bp) startup parameter 4–17

PRO command 2–10

PROAIW command 2–11

PROAPW command 2–12

PROBIW command 2–13

PROBRKR command 2–14

Procedure Editor startup commands
UNIX and Windows 2–3
Windows 2–5

Processes
cleaning up 2–23

PROLIB Memory (-plm) startup parameter 4–113

PROLIB Swap (-pls) startup parameter 4–114

PROQUIET Command 2–16

PROSERVE command 2–17

PROSHUT command 2–19

PROWDOG command 2–23

PROWIN command 2–24

Proxy Host (-proxyhost) startup parameter 4–118, 4–120

Proxy Password (-proxyPassword) startup parameter 4–119

Proxy Port (-proxyport) startup parameter 4–120

Proxy Userid (-proxyUserid) startup parameter 4–121

-proxyhost startup parameter 4–118, 4–120

-proxyPassword startup parameter 4–119

-proxyport startup parameter 4–120

-proxyUserid startup parameter 4–121

Q

-Q startup parameter 4–122

-q startup parameter 4–124

-Q2 startup parameter 4–123

Quick Request (-q) startup parameter 4–124

R

-r startup parameter 4–125

R-code In Code Page (-cprcodein) startup parameter 4–35

R-code Out Code Page (-cprcodeout) startup parameter 4–36

-rand startup parameter 4–126

Read-only (-RO) startup parameter 4–130

Record Buffer Size (-Mr) startup parameter 4–93

Record fragmentation 4–87

Recovery Log Threshold (-bithold) startup parameter 4–16

Remote databases
  starting a DataServer broker 2–14

Reread Nolock (-rereadnolock) startup parameter 4–127

-rereadnolock startup parameter 4–127

-rg startup parameter 4–129

-RO startup parameter 4–130

-rptint startup parameter 4–131

-rq startup parameter 4–133

-rr startup parameter 4–134

Run 4GL Client (-rg) startup parameter 4–129

Run Query Client (-rq) startup parameter 4–133

Run Run-time client (-rr) startup parameter 4–134

-rx startup parameter 4–135

S

-S startup parameter 2–21, 4–136

-s startup parameter 4–138

Save Temp Files (-t) startup parameter 4–152

Schema Cache File (-cache) startup parameter 4–22

Schema Field Cache Size (-fc) startup parameter 4–56

Search for files
  speeding up 4–124

Search path
  limiting 4–124

Secondary Login Broker (-m3) startup parameter 4–81

Secondary Login Broker server type 4–81

Segment Statistics (-yd) startup parameter 4–170

SELECT Pass Through Disable (-noSQLbyserver) startup parameter 4–102

Semaphore Sets (-semsets) startup parameter 4–139, 4–140

-semsets startup parameter 4–139, 4–140

Server Join (-nojoinbysqldb) startup parameter 4–101

Server startup parameters
  buffers 3–16
  networks 3–20
  performance 3–16
  server type specification 3–18
  statistic collection specification 3–19

Server-type startup parameters 3–18

Servers
  controlling number spawned for one protocol 4–90
  defined 1–2
  types 4–79, 4–80

Servers per Protocol (-Mp) startup parameter 4–90

-ServerType startup parameter 4–141

Service Name (-S) startup parameter 4–136
  PROSHUT 2–21

Shared memory
  overflow area 4–94

Shared-memory Overflow Size (-Mxs) startup parameter 4–94

Shutdown commands 2–19

Single-user Mode (-1) startup parameter 4–2

-Sn startup parameter 4–142

Sorting
  block size 4–154
  merge number parameter 4–155

Source code
  encrypted 4–135

Spawning servers
  manually 4–80

Speed Sort (-TB) startup parameter 4–154

Spin Lock Retries (-spin) startup parameter 4–143

-spin startup parameter 4–143

Spooling
  startup parameters 4–105

SQL-92 Stack Size (-SQLStackSize) startup parameter 4–145

-SQLStackSize startup parameter 4–145

-SQLStmtCacheSize startup parameter 4–146

Stack
  defined 4–138, 4–145
  overflow errors 4–138, 4–145

Stack Size (-s) startup parameter 4–138

Starting a DataServer broker 2–14

Starting Progress
  multi-user 2–9
    batch mode 2–8
    graphical mode 2–24
  multi-user server 2–17
  single-user 2–10
    batch mode 2–6
    graphical mode 2–24

Startup
  .pf file 1–3

Startup commands
  about 1–2
  syntax 2–2
  tasks you can perform 2–3

Startup parameters
  about 1–3, 3–2
  buffers 3–16
  changing defaults 3–2
  client connection 3–12
  client database connection 3–12
  client international 3–8
  client session parameters 3–2
  client session statistics 3–10
  database server 3–15
  database server international 3–18
  for spooling output 4–105
  memory 3–16
  networks 3–20
  performance 3–16
  server type specification 3–18
  statistic collection specification 3–19
  usage categories 3–2
  windows session 3–11

Startup Procedure (-p) startup parameter 4–109

Stash area
  defined 4–147

Stash Area (-stsh) startup parameter 4–147

Statement Cache Size (-SQLStmtCacheSize) startup parameter 4–146

Statistic Collection startup parameters 3–19

Statistics
  tracking access 4–11, 4–16

Statistics (-y) startup parameter 4–166, 4–167

Statistics with Cross-reference (-yx) startup parameter 4–172

Statistics with CTRL-C (-yc) startup parameter 4–169

Stream Code Page (-cpstream) startup parameter 4–37

-stsh startup parameter 4–147

-SV startup parameter 4–148

Syntax notation 1–xiii

System failures
  message for full-integrity Progress session 4–66
  message for no-integrity Progress session 4–66

T

-T startup parameter 4–150

-t startup parameter 4–152

Table Base (-basetable) startup parameter 4–13

Table Limit (-tablelimit) startup parameter 4–153

-tablerangesize startup parameter 4–153

-TB startup parameter 4–154

Temporary Directory (-T) startup parameter 4–150

Temporary files 4–152

Terminal Code Page (-cpterm) startup parameter 4–38

Thousands Separator (-numsep) startup parameter 4–104

Threshold Stall (-bistall) startup parameter 4–15

-TM startup parameter 4–155

-tok startup parameter 4–156

Token (-tok) startup parameter 4–156

-trig startup parameter 4–157

Trigger Location (-trig) startup parameter 4–157

Type of Server to Start (-ServerType) startup parameter 4–141, 4–145

Typographical conventions 1–xii

U

-U startup parameter 4–158

Usage categories
  startup parameters 3–2

Usage reporting
  -rptint startup parameter 4–131
  statistics with CTRL-C (-yc) parameter 4–169
  -usrcount startup parameter 4–160
  -y startup parameter 4–167

Use OS Locale (-useOsLocale) startup parameter 4–159

-useOsLocale startup parameter 4–159

User Count (-usrcount) startup parameter 4–160

User ID (-U) startup parameter 4–158

Users
  maximum number 4–96

-usrcount startup parameter 4–160

V

-v6colon startup parameter 4–162

-v6q startup parameter 4–163

Validation
  using dictionary and help 4–46

Version 6 Colon (-v6colon) startup parameter 4–162

Version 6 Query (-v6q) startup parameter 4–163

VLM Page Table Entry Optimization (-Mpte) startup parameter 4–92

W

-Wa -wpp startup parameter 4–164, 4–165

Watchdog process, see also PROWDOG utility

Wide area networks
  connecting 4–22

Windows Passthrough Printing (-Wa -wpp) startup parameter 4–164, 4–165

Windows session
  client startup parameters 3–11

Y

-y startup parameter 4–166, 4–167

-yc startup parameter 4–169

-yd startup parameter 4–170

Year
  two-digit 4–175

-yr4def startup parameter 4–171

-yx startup parameter 4–172

-yy startup parameter 4–175