

Progress
Version 9
Product Update Bulletin

© 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: 4501
Item Number: 89430;V91D

Contents

Preface . . . . . **xv**
Purpose xv
Audience xv
Organization Of This Manual xv
Typographical Conventions xvii
Syntax Notation xviii
Syntax Notation — SQL-92 xxi
Other Useful Documentation xxvi
Getting Started xxvi
Development Tools xxvii
Reporting Tools xxviii
4GL xxix
Database xxx
DataServers xxx
SQL-89/Open Access xxxi
SQL-92 xxxi
Deployment xxxii
WebSpeed xxxiii
Reference xxxiii

1. Overview Of New and Revised Features **1–1**
1.1 Changes In Version 9.1D 1–2
1.2 Changes In Version 9.1C 1–12
1.3 Changes In Version 9.1B 1–18
1.3.1 Progress On the Web 1–19
1.3.2 Administration 1–19
1.3.3 Installation 1–20
1.3.4 Progress Development Tools 1–20
1.3.5 Progress 4GL 1–23

1.3.6 Database 1–24
1.3.7 DataServers 1–24
1.3.8 AppServer 1–26
1.3.9 Open Client 1–27
1.3.10 Java Message Service (JMS) 1–28
1.3.11 Progress Documentation 1–28
1.4 Changes In Version 9.1A 1–29
1.4.1 Installation 1–30
1.4.2 Administration 1–31
1.4.3 Progress Development Tools 1–37
1.4.4 Progress 4GL 1–45
1.4.5 Database 1–53
1.4.6 DataServers 1–53
1.4.7 AppServer 1–55
1.4.8 Open Client 1–56
1.4.9 Internationalization 1–57
1.4.10 Progress SQL-92 1–62
1.4.11 XML 1–63
1.4.12 Sockets 1–65
1.4.13 Progress Version 9.1 Support For New Actuate APIs 1–69

2. Features By Book 2–1
2.1 Electronic Documentation 2–2
2.1.1 Additional Electronic Books 2–2
2.1.2 New Dynamic HTML Format 2–3
2.1.3 New PDF Format 2–3
2.1.4 Searching Across Multiple PDF Files 2–3
2.2 New Documents 2–4
2.2.1 New Documents For Version 9.1B 2–4
2.2.2 New Documents For Version 9.1A 2–5
2.3 Updated Documents 2–6
2.3.1 Updated Documents In Version 9.1D 2–6
2.3.2 Updated Documents In Version 9.1C 2–9
2.3.3 Updated Documents In Version 9.1B 2–11
2.3.4 Updated Documents In Version 9.1A 2–15
2.4 Books Retired As Of This Release 2–32

3. Feature Comparisons 3–1
3.1 Administration 3–2
3.1.1 AdminServer Security 3–2
3.2 Debug Alert (-debugalert) Startup Parameter 3–3
3.3 R-code Action Code Segments 3–4

3.4 Bit String Data Type Enhancements . . . 3–4
3.4.1 Maximum Length For VARBINARY . . . 3–5
3.4.2 Character Data Type Enhancements . . . 3–5
3.5 Databases . . . 3–6
3.5.1 Self-tuning -pwqdelay Parameter . . . 3–6
3.6 DataServers . . . 3–7
3.6.1 Single Executable For DataServer For ORACLE On UNIX . . . 3–7
3.7 Electronic Documentation . . . 3–8
3.7.1 PDF Search Capabilities In Version 9.1C . . . 3–8
3.8 Installation . . . 3–8
3.8.1 Changes In the Progress Core Installation Process For Progress Message Files . . . 3–8
3.9 Internationalization . . . 3–9
3.9.1 Modified Encoding Of the Numeric Format In the Trailer Of Table Dump (.d) Files . . . 3–10
3.9.2 Clarifications Concerning the Decimal Point and the Thousands Separator . . . 3–11
3.9.3 THAI Collation Of Code Page 620–2533 Renamed . . . 3–11
3.9.4 New Initial Value Segment No Convert (–isnoconv) Startup Parameter . . . 3–11
3.10 OCX Events . . . 3–12
3.10.1 Handling Of Asynchronous COM Events . . . 3–12
3.11 Progress Explorer Tool . . . 3–13
3.11.1 Starting the Progress Explorer Tool . . . 3–15
3.12 Progress 4GL Enhancements . . . 3–15
3.12.1 Copying MEMPTRs . . . 3–15
3.12.2 MEMPTRs As Parameters . . . 3–16
3.13 Progress SQL-92 . . . 3–17
3.13.1 New Progress SQL-92 JDBC Driver . . . 3–17
3.14 SmartObjects . . . 3–18
3.14.1 Dynamic SmartDataObject Super-procedure Changes . . . 3–18
3.14.2 New Naming Convention For SmartObject Files . . . 3–19
3.14.3 Multiple Field Sources In Smart Display Objects . . . 3–20
3.14.4 Stateless SmartDataObjects . . . 3–20
3.14.5 SmartDataViewer Changes . . . 3–21
3.14.6 SmartDataBrowser Scroll-bar Behavior Changes . . . 3–21
3.15 WebClient Enhancements . . . 3–22
3.15.1 Modifications To the WebClient Installer Process For Version 9.1D . . . 3–22
3.15.2 Automatic Conversion Of Configuration Files . . . 3–22

A. Progress DataServer A–1
A.1 Changes and Enhancements Made In Version 9.1D A–2
A.1.1 Progress Version 9.1D DataServer For ORACLE A–2
A.1.2 Progress Version 9.1D DataServer For Microsoft SQL Server A–6
A.1.3 Progress Version 9.1D DataServer For ODBC A–13
A.1.4 Progress/400 Product A–17
A.2 Changes and Enhancements Made In Version 9.1C A–18
A.2.1 Progress Version 9.1C DataServer For Microsoft SQL Server A–18
A.2.2 Progress Version 9.1C DataServer For ODBC A–19
A.2.3 Changes and Enhancements Made In Version 9.1B A–20
A.2.4 Progress DataServer For ORACLE A–20
A.2.5 Progress/400 DataServer A–21
A.2.6 Progress DataServer For Microsoft SQL Server A–22

B. R-code Features and Functions B–1
B.1 R-code Structure B–2
B.1.1 Factors That Affect R-code Size B–3
B.1.2 R-code File Segment Layout B–4
B.2 R-code Libraries B–6
B.3 R-code Libraries and PROPATH B–6
B.4 R-code Execution B–7
B.4.1 Standard R-code Execution Environment B–7
B.4.2 Memory-mapped R-code Execution Environment B–11
B.4.3 R-code Directory Management B–13
B.4.4 R-code Execution Environment Statistics B–14
B.5 R-code Portability B–17
B.6 Code Page Compatibility B–18
B.7 Database CRCs and Time Stamps B–18
B.7.1 Time Stamp Validation B–19
B.7.2 CRC Validation B–19
B.7.3 CRC Calculation B–20
B.7.4 CRC Versus Time Stamp Validation B–21
B.8 R-code CRCs and Procedure Integrity B–24
B.8.1 Assigning CRCs To Schema Triggers B–24
B.8.2 Validating CRCs For Schema Triggers B–25
B.8.3 RCODE–INFO Handle B–25

C. AppServer Internet Adapter C–1
C.1 AppServer Internet Adapter Architecture C–2
C.1.1 Using the AppServer Internet Adapter With HTTP C–4
C.1.2 Using the AppServer Internet Adapter With HTTPS C–5
C.1.3 AppServer Internet Adapter and Proxy Servers C–8
C.2 Installing and Licensing the AppServer Internet Adapter C–9
C.3 Installing and Configuring Web Servers and Java Servlet Engines C–10
C.4 Administering and Configuring an AppServer Internet Adapter C–13
C.4.1 ubroker.properties File C–13
C.5 Viewing AppServer Internet Adapter Connection and Configuration Information C–15
C.5.1 Connection Status C–16
C.5.2 Configuration Information C–17

D. Progress 4GL Language Reference D–1
ADM-DATA Attribute D–6
AMBIGUOUS Attribute D–6
ASYNCHRONOUS Attribute D–7
ASYNC-REQUEST-HANDLE Attribute D–7
BASE-ADE Attribute D–8
CALL-NAME Attribute D–10
CALL Object Handle D–11
CALL-TYPE Attribute D–22
Chained Widget Attributes D–23
CLEAR() Method D–24
CREATE CALL Statement D–24
CREATE SAX-READER Statement D–25
DEBUG-ALERT Attribute D–26
DEFINE BROWSE Statement D–27
DISABLE-DUMP-TRIGGERS( ) Method D–27
DISABLE-LOAD-TRIGGERS( ) Method D–28
ENABLE-CONNECTIONS( ) Method D–29
EVENT-PROCEDURE Attribute D–31
EVENT-PROCEDURE-CONTEXT Attribute D–31
FIND-CURRENT( ) Method D–32
FIND-FIRST Method D–33
FIND-LAST Method D–35
FIND-UNIQUE Method D–36
FIT-LAST-COLUMN Attribute D–38
GET-INDEX-BY-NAMESPACE-NAME() Method D–40
GET-INDEX-BY-QNAME() Method D–40
GET-LOCALNAME-BY-INDEX() Method D–41
GET-QNAME-BY-INDEX() Method D–41
GET-SOCKET-OPTION( ) Method D–42

GET-TYPE-BY-INDEX() Method D–44
GET-TYPE-BY-NAMESPACE-NAME() Method D–44
GET-TYPE-BY-QNAME() Method D–45
GET-URI-BY-INDEX() Method D–45
GET-VALUE-BY-INDEX() Method D–46
GET-VALUE-BY-NAMESPACE-NAME() Method D–46
GET-VALUE-BY-QNAME() Method D–47
HANDLER Attribute D–47
HTML-CHARSET Attribute D–48
IN-HANDLE Attribute D–49
INPUT-VALUE Attribute D–51
INVOKE() Method D–52
IS-PARAMETER-SET Attribute D–54
LITERAL-QUESTION Attribute D–54
LOCATOR-COLUMN-NUMBER Attribute D–55
LOCATOR-LINE-NUMBER Attribute D–56
LOCATOR-PUBLIC-ID Attribute D–56
LOCATOR-SYSTEM-ID Attribute D–57
Logical Function D–57
MEMPTR-TO-NODE-VALUE() Method D–59
MIN-COLUMN-WIDTH-CHARS Attribute D–60
MIN-COLUMN-WIDTH-PIXELS Attribute D–61
NODE-VALUE-TO-MEMPTR() Method D–62
NO-EMPTY-SPACE Attribute D–63
NUM-ITEMS Attribute D–64
NUM-PARAMETERS Attribute D–65
PARSE-STATUS Attribute D–66
PERSISTENT Attribute D–67
PRIVATE-DATA Attribute D–67
PROXY-PASSWORD Attribute D–68
PROXY-USERID Attribute D–69
Quoter Function D–70
RETURN-VALUE Attribute D–73
RETURN-VALUE-DATA-TYPE Attribute D–73
SAX-PARSE() Method D–74
SAX-PARSE-FIRST() Method D–75
SAX-PARSE-NEXT() Method D–76
SAX-ATTRIBUTES Object D–77
SAX-READER Object D–80
SCHEMA-PATH Attribute D–82
SERVER Attribute D–83
SET-INPUT-SOURCE() Method D–84
SET-PARAMETER() Method D–85
SET-SOCKET-OPTION( ) Method D–87
STOP-PARSING() Method D–90

SUPPRESS-NAMESPACE-PROCESSING Attribute D–91
TYPE Attribute D–92
UNIQUE-ID Attribute D–92
VALIDATION-ENABLED Attribute D–92
VIEW-FIRST-COLUMN-ON-REOPEN Attribute D–93
XML-SCHEMA-PATH Attribute D–93
XML-SUPPRESS-NAMESPACE-PROCESSING Attribute D–94

E. SQL Explorer E–1
E.1 Changes Made In Version 9.1 Through Version 9.1C E–2
E.1.1 SQL Explorer Tool Updates E–2
E.1.2 Additional SQL Explorer Properties E–3
E.1.3 New Properties For GUI and Character Interfaces E–4
E.1.4 Properties For the Character Interface Only E–6

F. Administration F–1
F.1 Changes and Enhancements Made In Version 9.1D F–2
F.1.1 Progress Explorer Tool F–2
F.1.2 AdminServer Security Enhancement F–3
F.2 Changes and Enhancements Made In Version 9.1C F–8
F.2.1 Progress Splash Screen F–8

G. Progress WebClient G–1
G.1 Progress Version 9.1D WebClient Installer Process G–2
G.1.1 Version 9.1C Functionality G–2
G.1.2 Version 9.1D Functionality G–3
G.2 Proxy Authentication For WebClient G–5

H. SQL-92 . . . . . H–1
H.1 Changes Made In Version 9.1D H–2
H.1.1 Multi-threaded Progress SQL-92 Server H–3
H.1.2 Progress SQL-92 Support For Progress Sequences H–4
H.1.3 Change In Progress SQL-92 Server Locking H–11

I. Database Tools I–1
I.1 Adjusting the SQL Width Property Setting I–2

J. Dynamic Invoke J–1
J.1 4GL Elements Related To Dynamic Invoke J–2
J.2 Using Dynamic Invoke Only When Necessary J–7
J.3 Example 1 — Invoking an External Procedure Nonpersistently J–9
J.4 Example 2 — Running an External Procedure Persistently and One Of Its Internal Procedures J–10
J.5 Example 3 — Using a CALL Object Multiple Times J–11
J.6 Example 4 — Getting an Attribute J–13
J.7 Example 5 — Setting an Attribute J–13
J.8 Example 6 — Driving a Series Of Dynamic Invokes From a TEMP-TABLE J–14

K. Optional NameServer Enhancement K–1
K.1 Overview Of the Optional NameServer K–2
K.2 No NameServer On the Server K–3
K.3 No NameServer On the Client K–3
K.3.1 4GL Client To AppServer K–3
K.3.2 OpenClient To an AppServer K–4
K.3.3 WebClient To an AppServer K–4
K.3.4 4GL Client To a DataServer K–4
K.3.5 4GL Client To Sonic Adapter K–5
K.3.6 WebSpeed Messenger and the AppServer Internet Adapter . . K–5

L. Installation L–1
L.1 Optional AdminServer User-group Authorization and Authentication . . . L–2
L.2 Changes In the Core Installation Of Progress Messages Files L–2
L.3 Updates To the Installation Process For Windows Platforms L–3
L.4 WebClient Installation L–3
L.5 Supported UNIX Platform L–4

Figures

Figure A–1: Create/Modify Database Record For DataServer Schema Dialog Box A–3
Figure A–2: PROGRESS DB To Oracle Conversion Dialog Box A–4
Figure A–3: Delta df To Oracle Conversion Dialog Box A–5
Figure A–4: Create/Modify Database Record For MSS Schema Dialog Box A–6
Figure A–5: PROGRESS DB To MS SQL Server Conversion Dialog Box A–7
Figure A–6: Create/Modify Database Record For DataServer Schema Dialog Box A–14
Figure A–7: PROGRESS DB To ODBC Conversion Dialog Box A–15
Figure B–1: R-code File Segment Layout B–5
Figure B–2: Standard R-code Execution Environment B–8
Figure B–3: Memory-mapped R-code Execution Environment B–12
Figure B–4: Progress Client Startup Options For –yd B–14
Figure B–5: Execution Buffer Map For –yd B–15
Figure B–6: Accessing the Session Sort File For –yd B–15
Figure B–7: Procedure Segment Information For –yd (Part 1) B–16
Figure B–8: Procedure Segment Information For –yd (Part 2) B–17
Figure C–1: AppServer Internet Adapter Architecture C–3
Figure C–2: AppServer Internet Adapter Architecture With HTTPS C–6
Figure C–3: AIA and HTTP Proxy Server C–8
Figure C–4: AIA Client Connection Information C–16
Figure C–5: AIA Configuration Information C–17
Figure G–1: Welcome Dialog Box - Maintenance Mode G–2
Figure G–2: Welcome To the WebClient Patch Install Program Dialog Box G–3
Figure G–3: SetUp Status Dialog Box G–4
Figure G–4: WebClient Patch Installation Complete Dialog Box G–4
Figure G–5: Information Dialog Box G–5
Figure H–1: Multi-threaded Progress SQL-92 Server H–3

Tables

Table 1–1: New Startup Parameters To Support the AppServer 1–26
Table 1–2: New Startup Parameters Since Version 9.0 1–35
Table 1–3: Obsolete Startup Parameters As Of Version 9.1 1–36
Table 1–4: Collations Added To Version 9.1 1–60
Table 1–5: Code-page Conversions Added To Version 9.1 1–61
Table 2–1: Progress Language Reference Updates In Version 9.1B 2–13
Table 2–2: Changed Book Titles 2–15
Table 2–3: New Attributes 2–21
Table 2–4: New Events 2–23
Table 2–5: New Functions 2–23
Table 2–6: New Object Handles 2–23
Table 2–7: New Methods 2–23
Table 2–8: New Options 2–25
Table 2–9: New Statements 2–25
Table 2–10: Modified Attributes 2–26
Table 2–11: Modified Events 2–27
Table 2–12: Modified Functions 2–27
Table 2–13: Modified Handles 2–27
Table 2–14: Modified Methods 2–28
Table 2–15: Modified Options 2–28
Table 2–16: Modified Phrases 2–28
Table 2–17: Modified Statements 2–28
Table 3–1: UNIX DataServer For ORACLE Executables 3–7
Table B–1: R-code Segments B–2
Table B–2: Metaschema Fields In CRC Calculation B–20
Table C–1: Qualified Web Servers and Java Servlet Engines C–11
Table D–1: CALL Object Attributes D–12
Table D–2: CALL Object Methods D–16
Table D–3: Keyword Constants For the CALL-TYPE Attribute D–22
Table D–4: Socket Connection Parameters D–29
Table D–5: Options For GET-SOCKET-OPTION() D–42
Table D–6: What IN-HANDLE Indicates and Who Sets It D–49
Table D–7: What INVOKE() Does For Each Call Type D–52
Table D–8: Values Of the SAX-READER's PARSE-STATUS Attribute D–66
Table D–9: Attributes Of the SAX-ATTRIBUTES Object D–77
Table D–10: Methods Of the SAX-ATTRIBUTES Object D–78
Table D–11: SAX-READER Object Attributes D–80
Table D–12: Methods Of the SAX-READER Object D–82
Table D–13: Options For SET-SOCKET-OPTION() D–88
Table J–1: Attributes Of the CALL Object Handle J–2
Table J–2: Methods Of the CALL Object Handle J–6
Table J–3: Invoking hello.p Dynamically and Statically J–7

Procedures

p-trload.p B–27
p-diexm1.p J–9
p-diexm2.p J–10
p-diexm3.p J–12
p-diexm4.p J–13
p-diexm5.p J–13
p-diexm6.p J–14

Preface

Purpose

This bulletin provides a brief description of the new andmodified features of Progress Version 9.1, including Version 9.1D. It guides you to where you can find more detailed information on the changes in the documentation set.

Audience

This bulletin is written for Progress application developers and system administrators who are upgrading their license to Progress Version 9.1D from an earlier release of Progress.

Organization Of This Manual

Chapter 1, "Overview Of New and Revised Features"

Summarizes the product features that are new or modified since Version 9.0B. Changes are listed by the Version 9.1 release during which they occurred. Entries include links to further information in the full documentation set.

Chapter 2, "Features By Book"

Notes changes to the way Progress documentation is supplied. This chapter also provides a list of the books that were added, updated, or retired in the Progress documentation set for each Version 9.1 release.

Chapter 3, "Feature Comparisons"

Describes how changed features in Version 9.1 might affect application code written in previous versions of Progress.

Appendix A, “Progress DataServer”

Describes Version 9.1 enhancements to the Progress DataServers for ORACLE, Microsoft SQL, and ODBC.

Appendix B, “R-code Features and Functions”

Describes the structure and management of Progress r-code, the executable code into which Progress compiles 4GL procedures. It also describes techniques for tuning r-code size and performance, and the use of time stamps and cyclic redundancy checks (CRCs) to maintain r-code and database integrity.

Appendix C, “AppServer Internet Adapter”

Describes the architecture of the AppServer Internet Adapter (AIA) and how to use the AIA to develop and deploy distributed Progress applications over the Web.

Appendix D, “Progress 4GL Language Reference”

Contains a list of the 4GL language elements that were added or modified in Progress Version 9.1D. This appendix also documents the 4GL language elements that were added in Version 9.1D including the SAX elements.

Appendix E, “SQL Explorer”

Describes enhancements to the Progress SQL Explorer tool

Appendix F, “Administration”

Describes enhancements to the Progress Explorer tool and enhancements to AdminServer security.

Appendix G, “Progress WebClient”

Describes new WebClient installation procedures and Proxy authentication for WebClient.

Appendix H, “SQL-92”

Describe the new SQL-92 features for the multi-threaded feature, Progress sequences, and the server locking feature.

Appendix I, “Database Tools”

Describes enhancements to the Database tools.

Appendix J, “Dynamic Invoke”

Describes how to invoke code dynamically.

Appendix K, “Optional NameServer Enhancement”

Describes the enhanced functionality that allows users to optionally specify no NameServer.

Appendix L, “Installation”

Describes the new and enhanced features of the installation and configuration processes for Windows and UNIX platforms.

Typographical Conventions

This manual uses the following typographical conventions:

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

The following typographical conventions are used to represent keystrokes:

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

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

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

  CTRL-X

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

  ESCAPE H
  ESCAPE CURSOR-LEFT

Syntax Notation

The syntax for each component follows a set of conventions:

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

  In this example, ACCUM is a keyword:

  SYNTAX

  ```
  ACCUM aggregate expression
  ```

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

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

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

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

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

  SYNTAX

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

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

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

  SYNTAX

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

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

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

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

  SYNTAX

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

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

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

  SYNTAX

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

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

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

SYNTAX

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

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

SYNTAX

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

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

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

  SYNTAX

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

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

  SYNTAX

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

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

  SYNTAX

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

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

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

  SYNTAX

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

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

  SYNTAX

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

Syntax Notation — SQL-92

The syntax for each SQL-92 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 an expression or a statement.

  In this example, GRANT, RESOURCE, DBA, and TO are keywords:

  SYNTAX

```
GRANT { RESOURCE, DBA } TO user_name [, user_name ] ... ;
```

- 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 GRANT statement you must supply at least one *user_name*.

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

  In this syntax example the first *user_name* is required, and additional *user_name* specifications are optional:

  SYNTAX

  ```
  GRANT { RESOURCE, DBA } TO user_name [, user_name ] ... ;
  ```

  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 the GRANT example, you must specify either RESOURCE or DBA or both, and at least one *user_name*. Any *user_name* specifications after the first are optional:

  SYNTAX

  ```
  GRANT { RESOURCE, DBA } TO user_name [, user_name ] ... ;
  ```

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

  For example, this excerpt from an ODBC application invokes a stored procedure using the ODBC syntax { call *procedure_name* (*param*) }, where braces and parentheses are part of the language:

  SYNTAX

  ```
  proc1( param, "{ call proc2 (param) }", param);
  ```

- A vertical bar (**|**) indicates a choice. In the CREATE SYNONYM example you must specify a *table_name*, *view_name*, or *synonym* but you can only choose one. Note that in all SQL-92 syntax, if you specify the optional *owner_name* qualifier, there must not be a blank between the period separator and the *table_name* or *view_name* or *synonym*:

SYNTAX

```
CREATE [ PUBLIC ] SYNONYM synonym
 FOR [ owner_name.]{table_name |view_name |synonym } ;
```

In the DELETE FROM example, you must specify one of *table_name* or *view_name*:

SYNTAX

```
DELETE FROM [ owner_name.]{table_name |view_name }
  [ WHERE search_condition ] ;
```

- 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 the ORDER BY example, you must include one expression *(expr)* or column position *(posn)*, and you can optionally specify the sort order as ascending (ASC) or descending (DESC). You can specify additional expressions or column positions for sorting within a sorted result set. The SQL-92 engine orders the rows on the basis of the first *expr* or *posn*. If the values are the same, the second *expr* or *posn* is used in the ordering:

  SYNTAX

  ```
  ORDER BY { expr | posn } [ ASC | DESC ]
    [ , [ { expr | posn } [ ASC | DESC ] ] ... ]
  ```

  In the GRANT example, you must include one *user_name*, but you can optionally include more. Note that each subsequent *user_name* must be preceded by a comma:

  SYNTAX

  ```
  GRANT { RESOURCE, DBA } TO user_name [, user_name ] ... ;
  ```

- In many 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, CREATE VIEW is followed by several optional items:

  SYNTAX

```
CREATE VIEW [ owner_name.]view_name
  [ ( column_name [, column_name ] ... ) ]
  AS [ ( ] query_expression [ ) ] [ WITH CHECK OPTION ] ;
```

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 platform, 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 Application Development Environment — Getting Started (Windows only)

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

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

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

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

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

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

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

Progress Startup Command and Parameter Reference

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

Welcome to Progress (Hard copy only)

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

Development Tools

Progress ADM 2 Guide

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

Progress ADM 2 Reference

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

Progress AppBuilder Developer's Guide (Windows only)

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

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

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

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

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

Progress Debugger Guide

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

Progress Help Development Guide (Windows only)

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

Progress Translation Manager Guide (Windows only)

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

Progress Visual Translator Guide (Windows only)

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

Reporting Tools

Progress Report Builder Deployment Guide (Windows only)

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

Progress Report Builder Tutorial (Windows only)

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

Progress Report Builder User's Guide (Windows only)

A guide for generating reports with the Progress Report Builder.

Progress Results Administration and Development Guide (Windows only)

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

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

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

4GL

Building Distributed Applications Using the Progress AppServer

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

Progress External Program Interfaces

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

Progress Internationalization Guide

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

Progress Language Reference

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

Progress on the Web

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

Progress Programming Handbook

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

Database

Progress Database Design Guide

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

Progress Database Administration Guide and Reference

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

DataServers

Progress DataServer Guides

These guides describe how to use the DataServers to access non-Progress databases. They provide instructions for building the DataServer modules, a discussion of programming considerations, and a tutorial.

Each DataServer has its own guide as follows:

- *Progress/400 Product Guide*
- *Progress DataServer for Microsoft SQL Server Guide*
- *Progress DataServer for ODBC Guide*
- *Progress DataServer for ORACLE Guide*

MERANT ODBC Branded Driver Reference

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

SQL-89/Open Access

Progress Embedded SQL-89 Guide and Reference

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

Progress Open Client Developer's Guide

A guide that describes how to write, build, and deploy Java and ActiveX applications, and Java applets that run as clients of the Progress AppServer. This guide includes information about how to expose the AppServer as a set of Java classes or as an ActiveX server, and how to choose an Open Client distribution package fro run time.

Progress SQL-89 Guide and Reference

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

SQL-92

Progress Embedded SQL-92 Guide and Reference

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

Progress JDBC Driver Guide

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

Progress ODBC Driver Guide

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

Progress SQL-92 Guide and Reference

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

Deployment

Progress Client Deployment Guide

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

Progress Developer's Toolkit

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

Progress Portability Guide

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

WebSpeed

Getting Started with WebSpeed

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

WebSpeed Installation and Configuration Guide

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

WebSpeed Developer's Guide

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

WebSpeed Product Update Bulletin

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

Welcome to WebSpeed (Hard copy only)

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

Reference

Pocket Progress (Hard copy only)

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

Pocket WebSpeed (Hard copy only)

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

1

Overview Of New and Revised Features

This chapter describes new and revised features for the product areas of Version 9.1:

- Changes In Version 9.1D
- Changes In Version 9.1C
- Changes In Version 9.1B
- Changes In Version 9.1A

1.1 Changes In Version 9.1D

This section lists new and enhanced features in Version 9.1D. For more detailed information about a feature, you are referred to the appropriate appendix in this bulletin or other Progress documentation.

- **Administration —** Version 9.1D includes the following enhancements and changes to administration:
  - The Progress Explorer Tool now provides a check box option that allows you to choose a controlling NameServer to work with the Progress AppServer, the Progress SonicMQ Adapter, the WebSpeed Transaction Server, or the Progress DataServers.
  - Enhanced AdminServer security through user authentication and user authorization; and enhanced AdminServer logging, which allows you to see more descriptive information when a login to the Progress Explorer or a connection to the AdminServer fails.
  - Additionally, the SonicMQ Adapter General properties now include a check box that allows you to start an Adapter instance automatically when the AdminServer starts. There is also a new Pool Range property in the SonicMQ Adapter Server properties that you use to specify settings for the number of threads within a SonicMQ Adapter broker process.
  - If you are running Fathom for Management Standard Edition, you now see Fathom-specific fields in the Progress Explorer Tool.
  - The WebSpeed Transaction Server, AppServer, and SonicMQ Adapter broker configuration security has been enhanced and now contains user name and password combinations.

For more information, see:

Manual:	Chapter 3, “Feature Comparisons,” and Appendix F, “Administration” in this bulletin.
HTML Online Help:	Progress Explorer HTML Help

- **AdminServer Security**— In Progress Version 9.1D, the AdminServer has enhanced security that allows an administrator to optionally require authorization and authentication for the AdminServer with an optional logging capability. The AdminServer can now require group names that list authorized users, as well as user names and passwords that authenticate authorized users of the AdminServer. For more information on the security enhancements for the AdminServer, see the following documentation:

  For general information on the enhanced AdminServer security, see:

Manual:	Appendix F, "Administration" in this bulletin.

  For information on how the changes to AdminServer authorization and authentication affect the WebSpeed Transaction Server, see:

Manual:	*WebSpeed Installation and Configuration Guide*

  For information on how the changes to AdminServer authorization and authentication affect the SonicMQ 4GL Adapter, see:

Manual:	*Progress External Program Interfaces*.

  For information on how the changes to AdminServer authorization and authentication affect the Progress Installation, see:

Manuals:	*Progress Installation and Configuration Guide Version 9 for UNIX*. *Progress Installation and Configuration Guide Version 9 for Windows*.

  For enhanced AdminServer security startup parameter usage, see:

Manual:	*Pocket Progress*

  For information on how the changes to AdminServer authorization and authentication affect the Progress Explorer, see the Progress Explorer online help.

- **Database** — Version 9.1D includes the following changes to the database:
  - The addition of Schema Mover (`mvsch`), a new `PROUTIL` qualifier. For complete syntax and more information see:

    Manual: *Progress Database Administration Guide and Reference*

  - The ability to perform a selective binary dump and load. For more information, see:

    Manual: *Progress Database Administration Guide and Reference*

  - The addition of the RFUTIL qualifier AIMAGE AIOFF, which disables after-imaging for a database in multi-user mode without shutting down the database. For more information, see:

    Manual: *Progress Database Administration Guide and Reference*

  - The addition of the `PROUTIL` qualifier ENABLEPDR, which allows for database replication using Peer Direct. For more information, see:

    Manual: *Progress Database Administration Guide and Reference*

  - The addition of self-tuning asynchronous page writers (APWs). For more information, see:

    Manual: *Progress Database Administration Guide and Reference*

  - The ability to "pin" shared memory by using the new `-pinshm` startup parameter. For more information, see:

    Manual: *Progress Database Administration Guide and Reference* and *Progress Startup Command and Parameter Reference*

- **DataServers** — Version 9.1D provides the following enhancements:
  - DataServer for Oracle Collation Table Support
  - DataServer for Oracle Pro-to-ora Include Default Option
  - DataServer fro Oracle Alternative No NameServer Option
  - DataServer for Oracle Dynamic Find Feature
  - DataServer for Oracle Quoter Function
  - DataServer for Microsoft SQL Server Collation Table Support
  - DataServer for Microsoft SQL Server Alternative No NameServer Option
  - DataServer for Microsoft SQL Server Insert Trigger Support
  - DataServer for Microsoft SQL Server New Dynamic Find Feature
  - DataServer for Microsoft SQL Server New Quoter Function
  - DataServer for Microsoft SQL Server FOR FIRST and FOR LAST Performance
  - DataServer for Microsoft SQL Server New -Dsrv Startup Parameter Option
  - DataServer for Microsoft SQL Server New Block Cursor
  - DataServer for ODBC Collation Table Support
  - DataServer for ODBC Alternative No NameServer Option
  - DataServer for ODBC New Dynamic Find Feature
  - DataServer for ODBC New Quoter Function
  - DataServer for ODBC New -Dsrv Startup Parameter Option
  - DataServer for ODBC New Block Cursor

  For more information, see:

Manual:	Appendix A, “Progress DataServer” in this bulletin.

- **Dynamic Invoke**— Version 9.1D supports dynamic invoke, which lets you invoke logic dynamically.

  For more information on this enhancement, see:

Manual:	Appendix J, "Dynamic Invoke" in this bulletin. Appendix D, "Progress 4GL Language Reference" in this bulletin.

- **Optional NameServer**— In Progress Version 9.1D, the NameServer is an optional component for several Progress servers. Progress Explorer now provides a check box that lets you control whether or not to us a NameServer.

  Depending on the type of client, Progress provides specific functionality for selecting the no NameServer option when a client connects to a server.

  For an overview optional NameServer in Progress Version 9.1D and how this option changes feature functionality as compared to prior versions, see:

Manuals:	Appendix K, "Optional NameServer Enhancement" in this bulletin.

  For information on how the changes to NameServer affect the Progress installation and configuration, see:

Manuals:	*Progress Installation and Configuration Guide Version 9 for Windows*. *Progress Installation and Configuration Guide Version 9 for UNIX*.

  For more information on how clients connect with the WebSpeed Transaction server using no NameServer, see the following documentation:

Manual:	*WebSpeed Installation and Configuration Guide*.

For more information on how clients connect with Sonic Adapter using no NameServer, see the following documentation:

Manual:	*Progress External Program Interfaces*

The DataServers in Progress Version 9.1C used the Data Service (`-DataService`) option to indicate that the no NameServer is to be used. Now, in Progress Version 9.1D, an additional connection parameter Direct Connect (`-DirectConnect`) is available. For more information, see:

Manual:	Appendix A, "Progress DataServer" in this bulletin.

For more information on the Direct Connect (`-DirectConnect`) connection parameter, see the individual server documentation and:

Manual:	*Pocket Progress* *Progress Installation and Configuration Guide Version 9 for UNIX*. *Progress Installation and Configuration Guide Version 9 for Windows*. *Progress Explorer*

- **Installation** — New and revised features have been introduced to the installation and configuration processes to support Windows and UNIX platforms, respectively and includes the following:

  – The 9.1D installation process for Windows and UNIX platforms has been updated to support an optional AdminServer user-group authorization and authentication feature. This feature allows users to specify group authorization requirements to start the Admin Service. For more information, see:

Manual:	Appendix F, "Administration" in this bulletin.

– Changes in the Core Installation of Progress Messages Files to Accommodate Future Add On Product Installations. For more information, see:

Manual:	*Progress Installation and Configuration Guide Version 9 for UNIX*. *Progress Installation and Configuration Guide Version 9 for Windows*.

– The 9.1D installation process for Windows platforms has been updated to support a new Silent Installation Option for the Shared Network Installation Utility (NetSetup.) For more information, see:

Manual:	Appendix L, "Installation" in this bulletin. *Progress Installation and Configuration Guide Version 9 for Windows*.

– For WebClient, the patch procedure and user interface associated with the existing WebClient Installer program has been updated to provide users a clearer, more direct way to install and confirm the installation of a WebClient patch during the WebClient installation process.

For more information, see:

Manual:	Appendix G, "Progress WebClient" in this bulletin.

– With the release of 9.1D, the SUN Solaris Sparc 8 (64 Bit) UNIX platform is supported.

For more information, see:

Manual:	*Progress Installation and Configuration Guide Version 9 for UNIX*

- **Open Client** — enhancements in Version 9.1D include:
  - Open Client access to an AppServer over the Internet
  - Additional support for MEMPTR and TABLE-HANDLE parameter
  - The option of connecting directly to an AppServer instead of going through a NameServer

  For more information, see:

Manual:	*Progress Open Client Developer's Guide* Appendix F, "Administration" and Appendix K, "Optional NameServer Enhancement" in the bulletin.

- **Progress 4GL Language Reference** — Version 9.1D incorporates new and modified 4GL language elements:
  - **Browse Widget Attributes**: FIT-LAST-COLUMN Option, MIN-COLUMN-WIDTH-CHARS Attribute, MIN-COLUMN-WIDTH-PIXELS attribute, NO-EMPTY-SPACE option, VIEW-FIRST-COLUMN-ON-REOPEN attribute.
  - **Buffer Object Methods and Attributes**: AMBIGUOUS attribute, DISABLE-DUMP-TRIGGERS method, DISABLE-LOAD-TRIGGERS method, FIND-CURRENT method, FIND-FIRST method, FIND-LAST method, FIND-UNIQUE method
  - **Chained Widget attributes**
  - **Define Browse statement**: FIT-LAST-COLUMN option, NO-EMPTY-SPACE option
  - **Logical function**
  - **Quoter function**
  - **SESSION Attributes**: PROXY-PASSWORD attribute, DEBUG-ALERT, PROXY-PASSWORD attribute, PROXY-USERID attribute

For more information, see:

Manual:	Appendix D, "Progress 4GL Language Reference" in this bulletin.

- **SAX**— Version 9.1D supports the Simple API for XML (SAX)

  For more information, see:

Manual:	*Progress External Program Interfaces* Appendix D, "Progress 4GL Language Reference" in this bulletin.

- **SonicMQ Adapter**— Version 9.1D provides the following enhancements:
  - Flow-control Management
  - Discardable Messages
  - Acknowledge-and-Forward Support
  - HTTP/HTTPS Tunneling Support
  - Load-balancing Support
  - Fail-over Support
  - Direct Connect Parameter
  - AdminServer Security

  For more information, see:

Manual:	*Progress External Program Interfaces*

- **Sockets**— Version 9.1D provides the following enhancements:
  - Ability to specify the length of the pending-connection queue
  - Additional socket options

  For more information, see:

Manual:	*Progress External Program Interfaces* Appendix D, "Progress 4GL Language Reference" in this bulletin.

- **Startup Parameters**
  - New startup parameter `-baseADE` allows developers to specify where the client should look for the ADE code.
  - New startup parameter `-clientlog` allows an application to automatically write all error messages and warning message to the specified log file.

  For more information, see the following documentation:

Manual:	*Progress Startup Command and Parameter Reference*.

- **WebClient** — Version 9.1D has the following changes and enhancements:
  - Proxy authentication
  - Improved installation
  - The option of connecting directly to an AppServer instead of going through a NameServer

  For more information, see:

Manual:	Appendix G, "Progress WebClient", in this bulletin.

1.2 Changes In Version 9.1C

This section lists new and enhanced features in Version 9.1C. For a more detailed description of a feature, you are referred to the appropriate appendix in this bulletin or other Progress documentation bulletin. This section provides information about:

This section describes new and revised features in Version 9.1C and documents the differences between Version 9.1C and Version 9.1B. For a more detailed description of a feature, you are referred to the appropriate appendix in this bulletin or other Progress documentation.

- **Administration —** Version 9.1C includes the following changes to administration:
  - The Progress GUI client and WebClient now display a splash screen during startup. For more information, see the following documentation:

    Manual: Appendix F, "Administration" in this bulletin.

  - Several Progress startup parameters have been created or modified. For more information, see the following documentation:

    Manual: Chapter 3, "Feature Comparisons," and *Progress Startup Command and Parameter Reference*.

  - Progress now supports up to four action code segments for the main procedure in an r-code file. For more information, see the following documentation:

    Manual: Chapter 3, "Feature Comparisons," and Appendix B, "R-code Features and Functions," in this bulletin.

- **ADM2 and AppBuilder** — The manner in which the SmartDataBrowser's scroll-bar operates has been enhanced to include record fetching capabilities. The SmartB2BObject now includes limited support for DTD originals in XML mapping files.

  Manual: *Progress AppBuilder Developer's Guide, Progress ADM 2 Guide*, and Chapter 3, "Feature Comparisons," in this bulletin.

  See the *Progress Release Notes* and ADM2 source files for information on any further changes to the ADM2.

- **Database** — The Progress Relational Database Management System (RDBMS) now supports Network File Server (NFS) configurations. NFS configurations on Linux platforms are not supported. For more information, see the following documentation:

Manual:	*Progress Database Administration Guide and Reference*

- **DataServers** — Version 9.1C provides the following enhancements:

  – The DataServer (-Dsrv) startup parameter has several new options, PRGRS_ALWAYS_INDEX, PRGRS_CACHE_CONN, PRGRS_PROC_TRAN, PRGRS_STABLE_CURS, and PRGRS_WAIT_DELAY. These new options apply for the Progress DataServer for ODBC and the Progress DataServer for Microsoft SQL Server. For more information on these new options, see the following documentation:

Manual:	*Progress DataServer for ODBC Guide* and Appendix A, "Progress DataServer" in this bulletin.

  – The Enterprise versions of the Progress DataServer for ODBC and Progress DataServer for Microsoft SQL Server now include Version 3.7 of the MERANT ODBC drivers for all the supported ODBC data sources. For the Personal versions of the DataServers, these drivers must be purchased from the vendor. For information about these drivers, see the PDF file `installation-path\odbc\ODBCREF.PDF`.

  – There is a new schema migration option for Progress/400, "Load Data Definitions." For more information on this new option, see the following documentation:

Manual:	Appendix A, "Progress DataServer" in this bulletin.

- **Debug Alert Enhancements** — The DEBUG-ALERT attribute on the SESSION system handle was modified to support debug alert enhancements. This attribute reads the value you set using the Debug Alert (`-debugalert`) startup parameter. For more information on this enhanced startup parameter, see

- **Electronic Documentation** — The Progress and WebSpeed PDF files now allow you to search across the entire documentation set. For more information about this feature, see the following documentation:

Manual:	*Progress Electronic Documentation Installation and Configuration Guide*

- **Installation** — At the request of the DG-UX vendor, Progress Version 9.1B was the last release to be offered on the DG-UX operating system. DG-UX has been removed from the list of supported platforms for 9.1C. For more information on available platforms, see the following documentation:

Manual:	*Progress Installation and Configuration Guide Version 9 for UNIX*

- **Internet Component Framework (ICF)** — The ICFPARAMETER attribute on the SESSION system handle language element was created in Progress Version 9.1C to supply Internet Component Framework (ICF) procedures with ICF-related data.

  NOTE: This attribute reads the value you set using the ICF Parameter (`-icfparam`) startup parameter.

  For more information, see:

Manual:	*Progress Startup Command and Parameter Reference*

- **Java** — Progress has migrated to Javasoft 1.3. For more information, see the following documentation:

Manual:	*Progress Installation and Configuration Guide Version 9 for UNIX* and *Progress Installation and Configuration Guide Version 9 for Windows*

- **OCX** — Version 9.1C incorporates a change in when “asynchronous” OCX events are handled. For more information, see the following documentation:

Manual:	Chapter 3, “Feature Comparisons,” in this bulletin

- **Open Client** — Progress' migration to Javasoft 1.3 has changed some of the requirements for generating proxies with Proxygen. For more information, see:

Manual:	Chapter 3, "Feature Comparisons," in this bulletin

- **SportsPro Sample Application —** The sample application now contains several new modules that demonstrate techniques for using Progress' latest features. For more information, see:

Manual:	*Progress Application Development Environment — Getting Started*

- **WebClient** — The WebClient has been upgraded with IntelliStream™ technology; support for digital signatures; support for a built-in installer, in addition to the external installers already supported; and the WebClient Application Assembler, a greatly expanded version of the PROWCAPP Editor. For more information, see:

Manual:	*Progress on the Web*

- **WebClient 4GL Enhancements —** The following language elements were modified to support WebClient enhancements provided in Progress Version 9.1B and Version 9.1C:
  - ADD-NEW-FIELD( ) method on the Temp-table object handle
  - AVAILABLE attribute on the Buffer-field object handle
  - CONNECT method (AppServer) to support URLs
  - Data Types for MEMPTRs
  - Image Phrase to support URL pathnames
  - LOAD-ICON method to support URL pathnames
  - LOAD-IMAGE method to support URL pathnames
  - LOAD-IMAGE-DOWN method to support URL pathnames
  - LOAD-IMAGE-INSENSITIVE method to support URL pathnames
  - LOAD-IMAGE-UP method to support URL pathnames

- LOAD-MOUSE-POINTER method to support URL pathnames
- LOAD-SMALL-ICON method to support URL pathnames
- SEARCH function to support URL pathnames
- SERVER attribute on the CODEBASE-LOCATOR system handle
- SYSTEM-HELP statement to support HTML Help
- TYPE attribute on the X-document and X-noderef object handles

The following language elements were created to support WebClient enhancements provided in Progress Version 9.1B and Progress Version 9.1C:

- CLIENT-TYPE attribute on the SESSION system handle
- DECIMALS attribute on the Buffer-field object handle
- GENERATE-MD5 option on the COMPILE statement
- HAS-RECORDS attribute on the Temp-table object handle
- MIN-SIZE option on the COMPILE statement

The following language elements were created in Progress Version 9.1C to support WebClient enhancements for downloading a client application's codebase (that is, an application's files) stored on an AppServer or a Web server:

- CODEBASE-LOCATOR system handle
- APPSERVER-INFO attribute
- APPSERVER-PASSWORD attribute
- APPSERVER-USERID attribute
- END-USER-PROMPT attribute
- KEEP-CONNECTION-OPEN attribute
- KEEP-SECURITY-CACHE attribute
- LOCATOR-TYPE attribute
- NEEDS-APPSERVER-PROMPT attribute

- NEEDS-PROMPT attribute
- PERSISTENT-CACHE-DISABLED attribute
- URL attribute
- URL-PASSWORD attribute
- URL-USERID attribute

- **XML Enhancements —** The following language elements were modified in Progress Version 9.1C to support XML enhancements for using namespaces and loading XML documents from URLs:
  - CREATE X-DOCUMENT statement
  - CREATE X-NODEREF statement
  - LOAD( ) method
  - SUBTYPE attribute
  - X-document Object handle
  - X-noderef Object handle

  The following language elements were created in Progress Version 9.1C to support XML enhancements for using namespaces and loading XML documents from URLs:
  - CREATE-NODE-NAMESPACE( ) method
  - GET-ATTRIBUTE-NODE( ) method
  - INITIALIZE-DOCUMENT-TYPE( ) method
  - LOCAL-NAME attribute
  - NAMESPACE-PREFIX attribute
  - NAMESPACE-URI attribute

- NORMALIZE( ) method
- PUBLIC-ID attribute
- SET-ATTRIBUTE-NODE() method
- SYSTEM-ID attribute

For more information on these features, see:

Manual:	*Progress Language Reference*

1.3 Changes In Version 9.1B

This section describes new and revised features in Version 9.1B and documents the differences between Version 9.1A and Version 9.1B for the following product areas:

- Progress on the Web
- Administration
- Installation
- Progress Development Tools
- Progress 4GL
- Database
- DataServers
- AppServer
- Open Client
- JMS
- Progress Documentation

1.3.1 Progress On the Web

Version 9.1B provides support for deploying and enabling a Progress multi-tier application to operate over the World Wide Web. For more information this support, see the *Progress on the Web* manual. Changes to support this feature include:

- A new 4GL client, the Progress WebClient, and tools designed to help you deploy an application for end users to install and run over the Web
- Enhanced 4GL and AppServer support that allows a 4GL client to connect to a AppServer through a Web server and that allows a 4GL client to access image files residing on a Web server
- New and enhanced SmartObjects to support the building of Web-enabled applications and applications with Java connectivity
- A new 4GL to Java Message Service (JMS) API and an adapter to allow 4GL applications to exchange messages using the Progress SonicMQ JMS

1.3.2 Administration

Version 9.1B includes changes to administration:

- The Progress Explorer includes plug-ins to manage new unified broker products, the Progress AppServer Internet Adapter and the Progress SonicMQ Adapter. For more information, see:

Manual:	*Progress on the Web*

- A new Use OS Locale (–UseOsLocale) startup parameter provides a means to set certain localization options from existing Windows settings. For more information, see:

Manual:	*Progress Startup Command and Parameter Reference*.

1.3.3 Installation

Version 9.1B includes changes to enhance installation:

- Progress now supports Windows 2000 and Redhat Linux.
- If you are upgrading to a new version of Progress you can retain changes made to the previous version's `ubroker.properties` file. To do this, replace the default properties file in the installed properties directory with the old properties file. When you start the AdminServer for the first time, a properties file conversion utility automatically runs. This conversion utility includes support for the new properties of the SonicMQ Adapter.

  For more information on these features, see:

Manual:	*Progress Installation and Configuration Guide Version 9 for Windows*, *Progress Installation and Configuration Guide Version 9 for UNIX*, and *Progress on the Web*

1.3.4 Progress Development Tools

Version 9.1B includes changes to the following Progress development tools:

- ADM Tools
- AppBuilder
- Application Compiler
- PRO*Tools

ADM Tools

The following changes include new SmartObjects that support Internet connectivity:

- A Simple SmartContainer that has no run-time representation, useful when creating modules that require no user interface.
- A new, dynamic SmartDataObject that reduces the client-side memory requirement.
- A new SmartBusinessObject that integrates up to twenty SmartDataObjects, allowing comprehensive updates in a single server-side transaction.
- A new SmartB2BObject that transforms 4GL data to and from XML for use in business-to-business messaging.

- New SmartSender and SmartReceiver objects that you can extend to create and handle a variety of message types.
- A new SmartRouter object that distributes incoming documents for processing.
- New SmartProducer and SmartConsumer objects that provide a messaging interface to the Progress SonicMQ system.

For more information, see:

Manual:	*Progress on the Web*

The following changes include general enhancements to support the ADM 2:

- Changes in the behavior of three super-procedure routines to support the dynamic object
- New naming convention for SmartObject files
- New property ObjectName in all SmartObjects
- New instance property in the SmartDataBrowser
- The ability to source data on a per-field basis, for SmartDataViewers and SmartDataBrowsers connected to a SmartBusinessObject.

For more information, see:

Manual:	*Progress on the Web*, and Chapter 3, "Feature Comparisons" in this bulletin
Windows Online Help:	ADM Help

AppBuilder

The changes to AppBuilder in this release are

- Tool icons for the new ADM SmartObjects added to the Object Palette
- The ability to indirectly invoke the new XMS Mapping Tool applet by opening an XML schema or mapping file

  For more information, see:

Manual:	*Progress on the Web*

Application Compiler

The Application Compiler supports a new option to compile r-code to the minimum size allowable for execution. This more compact code eliminates r-code segment information needed only by the Application Debugger and the Open Client Toolkit. For more information on this feature, see:

Manual:	*Progress Application Development Environment — Getting Started*, and *Progress on the Web*
Windows Online Help:	Application Compiler Help

PRO*Tools

The PRO*Tools palette includes the following new or enhanced applets:

- The AppServer Partition PRO*Tool is changed to the Service Parameter Maintenance PRO*Tool for Version 9.1B. This enhanced applet supports partition types in addition to AppServers, currently including JMS Servers. The enhanced PRO*Tool also makes the partition name a logical name, separate from the name of the Application Service that it supports. For more information, see:

Manual:	*Progress on the Web*
Windows Online Help:	PRO*Tools Help

- PROWCAPP Editor PRO*Tool for Version 9.1B is a new applet that allows the creation of the PROWCAPP files used to deploy WebClient applications. For more information, see:

Manual:	*Progress on the Web*
Windows Online Help:	PRO*Tools Help

- XML Mapping PRO*Tool for Version 9.1B is a new applet useful when creating the mapping files required by the new SmartB2BObject. For more information, see:

Manual:	*Progress on the Web*
Windows Online Help:	PRO*Tools Help

1.3.5 Progress 4GL

Version 9.1B includes the following changes to the 4GL:

- **Enhanced 4GL to support Progress on the Web**:
  - Modified CONNECT method on the server object handle to support AppServer connection parameters specified as URL pathnames
  - Modified PROPATH to support URL pathnames for use by the SEARCH function.
  - Modified SEARCH function to support a URL pathname for the file argument
  - Modified LOAD–IMAGE, LOAD–IMAGE–DOWN, LOAD–IMAGE–UP, LOAD–IMAGE–SENSITIVE, and LOAD–ICON methods to support a URL pathname for the file argument
  - New MIN–SIZE option on the COMPILE statement to reduce r-code size

  For more information, see:

Manual:	*Progress on the Web*

- **Miscellaneous 4GL changes**:
  - Modified SYSTEM–HELP statement to support HTML Help
  - New CLIENT–TYPE attribute on the SESSION system handle
  - New AVAILABLE attribute and DECIMALS attribute on the buffer-field object handle
  - New HAS–RECORDS attribute and modified ADD–NEW–FIELD( ) method on the temp-table object handle

  For more information, see:

Manual:	*Progress Language Reference*

1.3.6 Database

Version 9.1B includes the following changes to the Progress Database:

- Building indexes with binary load
- A quick method for deleting the contents of storage areas

  For more information, see:

Manual:	*Progress Database Administration Guide and Reference*

1.3.7 DataServers

Version 9.1B includes the following changes to the Progress DataServers:

- Modified Progress DataServer for ORACLE
- New Progress DataServer for Microsoft SQL Server
- Modified Progress DataServer for ODBC

Changes To Progress DataServer For ORACLE

DataServer for ORACLE changes include support for:

- The RAW–TRANSFER statement to allow rows to be stored and retrieved as Progress fields or ORACLE columns
- A utility for bulk inserts of many rows
- The 4GL command GET PREVIOUS, which, after an index reposition, retrieves rows prior to the point to which you repositioned

For more information, see

Manual:	Appendix A, "Progress DataServer" in this bulletin

New DataServer For Microsoft® SQL Server™

Progress now supports a DataServer for Microsoft SQL Server. The DataServer for SQL Server currently supports the same functionality as the DataServer for ODBC. However, enhancements that apply only to the Microsoft SQL Server database might only be implemented in the DataServer for Microsoft SQL Server in future releases.

For more information, see:

Manual:	*Progress DataServer for Microsoft SQL Server Guide*

Changes To Progress DataServer For ODBC

DataServer for ODBC changes (also available with the new DataServer for Microsoft SQL Server) include enhanced support for:

- Look ahead cursors as the default method for retrieving rows from a data source when the NO–LOCK condition is set on a FOR EACH statement or other 4GL query
- Indexed repositioning with the REPOSITION 4GL statement that enables the DataServer to no longer have to traverse through an existing cursor in search of the reposition location

For more information, see:

Manual:	*Progress DataServer for ODBC Guide*

1.3.8 AppServer

The Progress Version 9.1B AppServer includes the following enhancements:

- A new AppServer Internet Adapter (AIA) component that supports development and distribution of applications over the Internet. The AIA is a Java servlet that executes within a Web server using a Java Servlet engine (JSE) and extends the AppServer architecture to support HTTP tunneling. HTTP tunneling allows clients to tunnel through firewalls.
- A new Secure AppServer using the AIA with Security (AIA/S). The AIA/S supports HTTP tunneling over the Secure Socket Layer (SSL) using HTTPS.
- A new –URL connection parameter for the CONNECT( ) method. This connection parameter allows you to specify several other AppServer connection parameters formatted as a single URL. This URL can specify three protocols:
  - HTTP — To connect a client to an AppServer using an instance of the AIA
  - HTTPS — To connect a client to an AppServer using an instance of the AIA/S
  - AppServer — To connect a client to an AppServer directly using TCP/IP as in previous versions of the AppServer.
- A new `ubroker.properties` file validation command, `aiaconfig`, to validate configurations for instances of the AIA that you create by editing the `ubroker.properties` file.
- Two new startup parameters that allow a 4GL client to connect to an AIA or AIA/S mediated AppServer through a local firewall. Table 1–1 describes these new startup parameters.

Table 1–1: New Startup Parameters To Support the AppServer

New Parameter	Syntax	Purpose
Proxy Host	–proxyhost *host-address*	Specifies the host name or IP address of an HTTP-based proxy server.
Proxy Port	–proxyport *port*	Specifies the port on which the HTTP-based proxy server is listening.

For more information, see:

Manual:	*Progress on the Web*

1.3.9 Open Client

Version 9.1B includes the following changes to Progress Open Client support:

- Changed or previously undocumented features and requirements
- New parameter types

Changed Features and Requirements

The following features and requirements for working with the Open Client Toolkit have changed or were previously undocumented:

- ActiveX data type mapping for TEMP–TABLE parameters
- Modified shortcut for running ProxyGen
- UNIX requirements to run Batch ProxyGen
- Requirements to deploy ActiveX proxies
- Requirements for setting the CLASSPATH

For more information, see:

Manual:	Chapter 3, “Feature Comparisons” in this bulletin

New Parameter Types

You can now pass the following 4GL data types and objects as parameters in Open Client procedures:

- MEMPTR parameters
- TABLE–HANDLE parameters

For more information, see:

Manual:	*Progress Open Client Developer's Guide*

1.3.10 Java Message Service (JMS)

Version 9.1B includes the following changes that support 4GL access to JMS:

- The SonicMQ Adapter, part of the AppServer Administration framework, communicates between clients and the SonicMQ message broker.
- The Progress SonicMQ Adapter comes with a 4GL-JMS API to provide access to Java Message Service (JMS) messaging from Progress 4GL applications.

For more information, see:

Manual:	*Progress External Program Interfaces*

1.3.11 Progress Documentation

Version 9.1B includes bug fixes and some content changes to both the online and hard-copy documentation since Version 9.1A. For more information, see Chapter 2, "Features By Book."

1.4 Changes In Version 9.1A

This section describes new and revised features in Version 9.1A and documents the differences between Version 9.0 and Version 9.1A for the following product areas:

- Installation
- Administration
- Progress Development Tools
- Progress 4GL
- Database
- DataServers
- AppServer
- Open Client
- Internationalization
- SQL-92
- XML
- Sockets
- Actuate APIs

1.4.1 Installation

This section summarizes changes to the Progress Version 9.1 Installation utility for the Windows and UNIX operating systems:

- The Progress Version 9.1 Electronic Documentation (EDOC) Installation utility now installs the DynaText 4.1 Viewer on Windows operating systems.
- The Installation utility now installs MERANT ODBC drivers during the installation process for Windows operating systems. In addition to the Progress SQL-92 ODBC driver (which were installed with Progress Version 9.0B), Progress Version 9.1 installs new MERANT ODBC drivers to enable the Enterprise DataServer for ODBC to access a number of supported data sources.

  NOTE: Progress Version 9.1 installs these additional drivers only if you install the Enterprise DataServer for ODBC.

- A new feature allows users to add components or subcomponents to existing Progress V9.1 and later installations without having to reinstall Progress.
- The Progress Installation utility now installs the Microsoft Management Console (MMC) on Windows operating systems if you are using one of the six language versions available with the Installation utility, and if you do not already have MMC 1.1 or later installed on your system. The Progress Explorer now operates as a snap-in component of the MMC. The Installation utility calls the Microsoft-supplied install program to install the MMC.
- The Progress Installation utility now installs Microsoft HTML Help Version 1.22 on Windows operating systems if you are using one of the six language versions available with the Installation utility, and if you do not already have Microsoft HTML Help Version 1.22 or later installed on your system. The Installation utility calls the Microsoft-supplied install program to install Microsoft HTML Help.
- The Progress Installation utility now installs Microsoft Java (MS JVM) Build 3177 or later if you do not already have MS JVM 3177 or later already installed on your system.
- Progress now provides a silent mode for the installation on UNIX platforms (this feature was already available on Windows platforms). You can use the batch install feature to launch the Progress installation from your own install utility, and Progress will be installed silently during your install. Batch install can be used only for Typical and Custom installations.

For more information, see:

Manual:	*Progress Installation and Configuration Guide Version 9 for Windows*, and *Progress Installation and Configuration Guide Version 9 for UNIX*

1.4.2 Administration

Progress Version 9.1 includes the following changes:

- The Progress Explorer tool, the graphical user interface that provides an easy way for you to manage Progress servers, is now a snap-in for the Microsoft Management Console (MMC). The Progress Explorer tool runs on the Windows 95, Windows 98, Windows 2000, and Windows NT platforms. This release provides online HTML help.
- New memory-mapped procedure library format
- New Progress initialization file options
- New and obsolete startup parameters

Progress Explorer Tool

The Progress Explorer tool, the graphical user interface that provides an easy way for you to manage Progress servers, is now a snap-in for the Microsoft Management Console (MMC). MMC is a general-purpose framework used for hosting system and network administration tools developed by Microsoft for the Windows platforms. When you launch Progress Explorer, you are actually launching an instance of MMC that has been preconfigured by Progress to contain the Progress Explorer snap-in. Much of the functionality available to you through the Progress Explorer tool is provided by MMC. For more information about MMC, see the MMC online help. For more information about the Progress Explorer tool, see:

Manual:	Chapter 3, "Feature Comparisons," in this bulletin.
HTML Online Help:	Progress Explorer HTML Help

New Memory-mapped Procedure Library Format

In addition to creating standard procedure libraries, the PROLIB utility now allows you to generate memory-mapped procedure libraries from standard procedure libraries. With standard procedure libraries, each client accesses and executes a run-time instance of r-code segments in local memory. Memory-mapped procedure libraries allow multiple clients on a single system to access and execute the same r-code segments in shared memory.

The following changes were made to the PROLIB utility to support memory-mapped procedure libraries:

- A new –makeshared parameter lets you generate a memory-mapped procedure library from a standard procedure library.
- In addition to existing library information, the –list parameter now displays the format of the library. The valid library formats are STANDARD and MAPPED.

  For more information, see:

Manual:	*Progress Programming Handbook,* and *Progress Client Deployment Guide*

New Progress Initialization File Options

You can now define the following options in the Progress initialization file:

- Define the following options in the Startup section of the Progress initialization file:
  - AlignFrameTitles lets you specify the alignment of frame titles in a graphical user interface. This option applies to graphical interfaces only.
  - CarefulPaint lets you set the accuracy with which Progress repaints overlapping 3-D widgets and frames. This option applies to graphical interfaces only.
  - Keep3DfillinBorder lets you display a disabled fill-in field with a border. This option applies to graphical interfaces only.

  - Opsys lets you specify the value used by the OPSYS preprocessor directive or the value returned by the OPSYS function. This option is supported in Windows only.
  - PopupAppOverridesSys lets you override the display of the system-default pop-up menu. This option applies to graphical interfaces only.
  - PopupNoSysDefault lets you suppress the system-default pop-up menu. This option applies to graphical interfaces only.
  - PopupOnMouseDown lets you display a pop-up menu when the right mouse button is depressed. This option applies to graphical interfaces only.
  - UseNative3D lets you use the 3D drawing capabilities in Windows 95/NT 4.0, or the 3D drawing capabilities in Progress. This option applies to graphical interfaces only.
  - V6FrameBGC lets you specify the default background color for all frames that use the V6FRAME option. This option applies to graphical interfaces only.
  - WindowSystem lets you specify the value used by the WINDOW–SYSTEM preprocessor directive. This option is supported in Windows only.

- Define the following options in the WinChar Startup section of the Progress initialization file:
  - BrowseRowMarker lets you specify a character to represent a browse row marker. This option applies to character clients in Windows only.
  - CursorHeight lets you specify the height of a cursor as a percentage. This option applies to character clients in Windows only.
  - MenuMnemonicColor lets you specify the color of mnemonic characters in menu items. This option applies to character clients in Windows only.
  - SingleLineBorder lets you specify the characters used to draw borders and lines (such as frame borders, rectangles, submenu and pop-up menu borders, and horizontal and vertical lines). This option applies to character clients in Windows only.
  - SysCheckmark lets you specify a character to represent a selection marker in widgets (such as a selection list, menu, or browser). This option applies to character clients in Windows only.

– UseDosCursor lets you change the cursor from an underscore to a solid block when switching from insert mode to over-strike mode in a fill-in field or editor widget. This option applies to character clients in Windows only.

For more information, see:

Manual:	*Progress Client Deployment Guide*

New and Obsolete Startup Parameters

Table 1–2 lists startup parameters that are new as of Version 9.1.

Table 1–2: New Startup Parameters Since Version 9.0

New Parameter	Syntax	Purpose
Async Queue Size	–asyncqueuesize *n*	Specifies the total number of bytes allowed for asynchronous request buffers allocated for a client.
Debug Alert Box	–debugalert	Adds a Help button to alert boxes that allows a user to obtain a stack trace.
Initial Value Segment No Convert	–isnoconv	Disables a code page conversion. Use if the conversion garbles preconverted international characters or causes the text segment to grow too large.
Fractional Separator	–numdec *numeric–value*	Changes the character that represents a number's decimal point in formatted text.
Thousands Separator	–numsep *numeric–value*	Changes the character that represents a number's thousands separator in formatted text.
Reread Nolock	–rereadnolock	Directs Progress to reread the record from the database, even if the record is already in a buffer.
Four Digit Year Default	–yr4def	Outputs a four-digit year from EXPORT, MESSAGE and PUT UNFORMATTED statements that might use a two-digit year.

For more information, see:

Manual:	*Progress Startup Command and Parameter Reference*

Table 1–3 lists startup parameters that are obsolete as of Version 9.1.

Table 1–3: Obsolete Startup Parameters As Of Version 9.1 *(1 of 2)*

Obsolete Parameter	Syntax	Purpose
Character Set	–charset	Identifies the code page that Progress uses in memory.
Cyclic Redundancy	–crc	Specifies using Cyclic Redundancy Checking to ensure integrity between the database schema and compiled code.
Distributed Transaction Manager	–decdtm	Allows two-phase commit on a VMS system.
Heap Size	–hs	Specifies the heap space on System V UNIX machines that use shared memory.
Network Version	–Nv	Specifies the version number of the connection protocol.
Page Writer Queue Delay	–pwqdelay	Determines how often the asynchronous page writer (APW) scans the page writer queue.
Page Writer Queue Minimum	–pwqmin	Determines the minimum number of buffers required on the page writer queue before the APW writes them to disk
Page Writer Scan	–pwscan	Determines the the number of buffers that an APW scans during a buffer scan cycle.
Page Writer Scan Delay	–pwsdelay	Determines the number of seconds delay between each APW scan of the buffers
Page Writer Maximum Buffers	–pwwmax	Determines the maximum number of modified buffers to write to disk during a buffer scan cycle
Stream	–stream	Specifies the code page for stream I/O.

Table 1–3: Obsolete Startup Parameters As Of Version 9.1 *(2 of 2)*

Obsolete Parameter	Syntax	Purpose
SYBASE	–SYBt	Sets the time of a SYBASE client time out in seconds.
Time Stamp	–tstamp	Uses time stamp information rather than cyclic redundancy (CRC) to enforce consistency between r-code and a schema
Video Write	–vw	Allows direct video write compatibility when running Progress from a PC.
Extended Alphabet Support	–xc	Specifies the collation table within the `convmap.cp` file.

For more information, see:

Manual:	*Progress Startup Command and Parameter Reference*

1.4.3 Progress Development Tools

The following sections describe new or revised development tools in Progress Version 9.1.

ADM Tools

The following Progress ADM Tools are new or revised in Progress Version 9.1:

- **SmartDataObject** — Three significant revisions have been made to SmartDataObjects in Version 9.1:
  - You can define temp-tables for SmartDataObjects that contain subsets of the fields in the database tables that they will access.
  - They can run statelessly.
  - They now have four transaction-level validation entry points instead of only one.

For more information, see:

Manual:	*Progress ADM 2 Guide*, *Progress ADM 2 Reference*, and *Progress AppBuilder Developer's Guide*

Dynamic SmartDataBrowser — This is a new SmartObject that can be configured dynamically to browse and update the query of any SmartDataObject. This dynamic object will, in many cases, eliminate the need to build a specific SmartDataBrowser or set of SmartDataBrowsers for each SmartDataObject. For more information, see:

Manual:	*Progress ADM 2 Guide*, *Progress ADM 2 Reference*, and *Progress AppBuilder Developer's Guide*

- **SmartDataViewer** — In Version 9.1, the SmartDataViewer template `viewer.w` has been revised to allow you to drop other SmartObjects onto a SmartDataViewer instance; that is, a SmartDataViewer can now be a container. However, you cannot drop all types of objects onto a SmartDataViewer—it is intended to contain only basic objects or SmartDataFields. (See the "SmartDataField" item for a description of this new SmartObject.) For more information, see:

Manual:	*Progress ADM 2 Guide*, *Progress ADM 2 Reference*, and *Progress AppBuilder Developer's Guide*

- **SmartDataField** — This new SmartObject, which displays a single data field, is intended for inclusion only in a SmartDataViewer. You typically use a SmartDataField to provide a specially formatted display of the data in a selected field in the containing SmartDataViewer. You can create any user interface for a SmartDataField as long as it meets certain basic requirements described in the documentation for this SmartObject. Note that the ADM provides one specially formatted SmartDataField called a SmartSelect. (See the "SmartSelect" item for a description of this new SmartObject.) For more information, see:

Manual:	*Progress ADM 2 Guide*, *Progress AppBuilder Developer's Guide*

- **SmartFolder** — In Version 9.1, you can set Page 0 in a SmartFolder. When Page 0 visualizes, either at design time or at run time, it simply provides a blank page that hides all objects not on Page 0. For more information, see:

  Manual: *Progress ADM 2 Guide*, *Progress ADM 2 Reference*
  Progress AppBuilder Developer's Guide

- **SmartFilter** — This is a new SmartObject that receives selection criteria for a SmartSelect. (See the "SmartSelect" item for a description of this new SmartObject.) It displays data fields in a frame, filtered by user-specified selection criteria. It follows the query-by-example model and is implemented as a dynamic SmartObject; that is, the selection criteria (the filter) are specified at run time rather than at design time. The application user specifies selection criteria in the filter fields, using Apply, Blank, and Reset radio buttons as needed. For more information, see:

  Manual: *Progress ADM 2 Guide*, *Progress ADM 2 Reference*
  Progress AppBuilder Developer's Guide

- **SmartToolbar** — This is a new SmartObject that provides a toolbar and/or menu interface to application commands that perform various actions; for example, running other windows or procedures or performing navigation and transaction actions in the current object. This object makes it possible to provide toolbar-type functionality directly rather than by using a Navigation SmartPanel and an Update SmartPanel, collectively, as a sort of toolbar, as in Progress Version 9.0. For more information see:

  Manual: *Progress ADM 2 Guide*, *Progress ADM 2 Reference*
  Progress AppBuilder Developer's Guide

- **SmartSelect** — This new SmartObject is a special type of SmartDataField that performs a lookup on an associated SmartDataObject field; that is, it retrieves and displays a predefined set of values from which an application user can choose a single value. Inherited SmartDataField behavior ensures that the SmartSelect automatically displays the current field values and that modified SmartSelect values are passed back to the associated SmartDataObject field. Like the basic SmartDataField, the SmartSelect is intended for inclusion only in a SmartDataViewer. For more information, see:

  Manual: *Progress ADM 2 Guide*, *Progress ADM 2 Reference*
  Progress AppBuilder Developer's Guide

- **Extending the ADM** — Version 9.1 provides two new tools for developing extensions to the ADM; that is, for extending the ADM class hierarchy with new or customized classes. (In Version 9.0, you do this by copying and modifying existing class files.) The two new tools are:

  – Custom class files. The ADM provides a set of these files for each ADM class. You use them as the basis for a customized ADM class that is modeled on an existing class.

  – The New ADM Class tool. This AppBuilder tool creates a set of of ADM class files (standard and custom) according to your specifications. You use the standard class files that the tool creates as the basis for a new ADM class.

  While each of these new ADM features provides a set of class files for a customized or new class, these class files do not completely define the class and cannot be used as is. You must modify them with the application logic that drives the class: properties, super procedures, and so forth. For more information, see:

  Manual: *Progress ADM 2 Guide* and *Progress ADM 2 Reference*

AppBuilder

The AppBuilder now has the following new or enhanced features:

- **Advanced Editing Features** — If your Progress installation runs under a development license, AppBuilder editors include the following advanced editing features:
  - Color coding
  - Embedded color coding
  - Syntax expansion
  - Alias expansion
  - Syntax indenting
  - SmartPaste

  NOTE: The advanced editing features are enabled by default. However, the Progress advanced editing features do not support bidirectional typing on Hebrew and Arabic Windows operating systems. You can turn the advanced editing features on or off to meet your needs.

- **Section Editor** — You now have the option to associate a separate window with each open workspace. You can also continue to use a single window that shifts context according to which workspace has focus.

- **The Section Editor —** Now allows you to leave open the windows from which you paste certain tokens into your source code.

- **The File menu —** Now includes the following:
  - A Save All option that you use to save structured procedures. Structured procedures are those procedures being maintained by AppBuilder.
  - A Most Recently Used file list option. You can enable or disable this option. If you enable this option, you can use the File drop-down menu in the AppBuilder menu bar to select and open a file from a list of most recently used files. You use the Options→ Preferences option on the AppBuilder menu to specify the number of files you want to appear in this list.
  - A Print option that allows you to print either the section that is current in the Section Editor, or the entire current procedure. Qualities such as page length and immediate versus prompted printing are Preference items.

- **The Tools menu** — Includes an item (ADM) that, when selected, opens a dialog box to begin the process of creating a new ADM class.
- **The Objects Palette** — Now has several additional tools:
  - SmartFolder, implements the popular tabbed-file-folder metaphor
  - Dynamic SmartDataBrowser, offers easier configuration (right-click on the SmartDataBrowser tool)
  - SmartDataField, brings ADM sophistication down to the field level
  - SmartSelect, a data-driven list (right-click the SmartDataField tool)
  - SmartFilter, a query-by-form object addressed to end users of your applications
  - SmartToolbar, offers SmartPanel functionality in a more integrated way
  - SmartContainer, allows you to conveniently create parent/child relationships between two organizer objects that would normally be independent
- **The SmartDataObject instance properties** —Now include a checkbox for forcing state-aware (persistent-connection) operation. The default mode is stateless (connectionless) unless the AppServer is configured to run with bound connections.
- **Database Field property sheet** — This property sheet now displays the label and format information when you add a Database field to a window. The label and format information is provided using the Data Dictionary. If you want to edit these values, you can use the Database Field button in the property sheet.
- **Supported graphic-file formats** — In addition to the previously supported .ICO and .BMP graphic-file formats, AppBuilder now supports these graphic-file formats: .CAL, .CUT, .DCX, .DIB, .EPS, .GIF, .ICA, .IFF, .IMG, .JPG, .LV, .MAC, .MSP, .PCD, .PCT, .PCX, .PSD, .RAS, .TGA, .TIF, .WMF, .WPG, .XBM, .XPM.

For more information, see:

Manual:	*Progress AppBuilder Developer's Guide*
	Progress Application Development Environment — Getting Started

Application Debugger

When you run the Application Debugger, you can now use the SESSION handle during a debugging session to include a Help button in alert boxes. The Help button provides access to a Stack Trace dialog box that displays 4GL stack trace information for the currently executing Progress session. For more information, see:

Manual:	*Progress Debugger Guide*

Procedure Editor

The Procedure Editor has the following new or enhanced features:

- **Advanced Editing Features** — If your Progress installation runs under a development license, the Procedure Editor includes the following Advanced Editing Features:
  - Color coding
  - Embedded color coding
  - Syntax expansion
  - Alias expansion
  - Syntax indenting
  - SmartPaste

  NOTE: The advanced editing features are enabled by default. However, the Progress advanced editing features do not support bidirectional typing on Hebrew and Arabic Windows operating systems. You can turn the advanced editing features on or off to meet your needs.

- **Most Recently Used File list** — The Procedure Editor File menu now has a Most Recently Used File list option. You can enable or disable this option. If you enable this option, you can use the File drop-down menu in the Procedure Editor menu bar to select and open a file from a list of most recently used files. You use the Options→ Preferences option on the AppBuilder menu to specify the number of files you want to appear in this list.

For more information, see:

Manual:	*Progress AppBuilder Developer's Guide* *Progress Application Development Environment — Getting Started*
Windows Online Help:	Procedure Editor Help

Pro*Tools

The PRO*Tools palette includes the following new or enhanced applets:

- **PRO*Spy** for Version 9 — **A** new applet for debugging applications containing ADM2 SmartObjects. For more information, see the following documentation:

Windows Online Help:	PRO*Tools Help

- **The AppServer Session Information PRO*Tool** — A new applet that provides information about a running AppServer session defined with the AppServer Partition Deployment PRO*Tool. For more information, see:

Manual:	*Building Distributed Applications Using the Progress AppServer*
Windows Online Help:	PRO*Tools Help

- **The Procedure Object Viewer** — Now lists any persistent super procedures associated with a selected persistent procedure. For more information, see the following documentation:

Windows Online Help:	PRO*Tools Help

- **The Database Connections PRO*Tool** — Is now modal and displays enhanced schema information, including table, field, and index details. For more information, see the following documentation:

Windows Online Help:	PRO*Tools Help

Windows Help

Progress Windows help for Version 9.1 is modular. All installed Windows help systems appear in a combined table of contents, index, and full text search. Windows help systems that are not installed on a user's computer do not appear in the combined table of contents, index, or full text search. For more information, see the following documentation:

Windows Online Help:	Progress Help

1.4.4 Progress 4GL

Progress Version 9.1 includes the following 4GL enhancements:

- **New asynchronous request object** and the following new and modified 4GL elements that support it:
  - New ASYNC–REQUEST–COUNT attribute
  - New CANCELLED attribute
  - New CANCEL–REQUESTS( ) method
  - New COMPLETE attribute
  - Modified DISCONNECT( ) method
  - New ERROR attribute
  - New EVENT–PROCEDURE attribute

  - New EVENT–PROCEDURE–CONTEXT attribute
  - New FIRST–ASYNC–REQUEST( ) method
  - New LAST–ASYNC–REQUEST( ) method
  - Modified NAME attribute
  - Modified NEXT–SIBLING attribute
  - New PERSISTENT–PROCEDURE attribute
  - Modified PREV–SIBLING attribute
  - Modified PRIVATE–DATA attribute
  - New PROCEDURE–COMPLETE event
  - New PROCEDURE–NAME attribute
  - New QUIT attribute
  - Modified RUN statement
  - Modified SELF system handle
  - Modified SERVER attribute
  - Modified server object handle
  - New STOP attribute
  - Modified TYPE attribute
  - Modified WAIT–FOR statement
- New temp-table object and the following new and modified 4GL elements that support it:
  - New CREATE TEMP–TABLE statement
  - New CLEAR( ) method
  - New TEMP–TABLE–PREPARE( ) method
  - New CREATE–LIKE( ) method
  - New ADD–FIELDS–FROM( ) method

  - New ADD–NEW–FIELD( ) method
  - New ADD–LIKE–FIELD( ) method
  - New ADD–NEW–INDEX( ) method
  - New ADD–INDEX–FIELD( ) method
  - New ADD–LIKE–INDEX( ) method
  - Modified NAME attribute
  - New DEFAULT–BUFFER–HANDLE attribute
  - New UNDO attribute
  - New PREPARED attribute
  - New PRIMARY attribute
  - Modified DEFINE PARAMETER statement

- **Buffer object and Buffer-field object** — Includes enhancements to support the temp-table object:
  - Modification to CREATE BUFFER statement to support using a temp-table as a table-name
  - New TABLE–HANDLE attribute
  - Modified NEXT–SIBLING attribute
  - New BUFFER–COPY( ) method
  - New BUFFER–COMPARE( ) method
  - New INDEX–INFORMATION( ) method
  - New RAW–TRANSFER( ) method
  - New PRIMARY attribute
  - Modified VALIDATE–EXPRESSION attribute
  - New FIRST–BUFFER attribute

- **Super procedure enhancements** — Support modifications of the super procedure search order:
  - Modified ADD–SUPER–PROCEDURE( ) method
- **Enhancements to support a dynamic browse**:
  - New CREATE BROWSE statement
  - New AUTO–VALIDATE attribute
  - Modified DYNAMIC attribute
  - Modified QUERY attribute
  - Modified MULTIPLE attribute
  - New ADD–LIKE–COLUMN( ) method
  - New ADD–COLUMNS–FROM( ) method
  - New ADD–CALC–COLUMN( ) method
  - New GET–BROWSE–COLUMN( ) method
  - New BUFFER–FIELD attribute
  - Modified SCREEN–VALUE attribute
- **Enhancements for browse column visibility**:
  - Modified VISIBLE attribute
  - New NUM–VISIBLE–COLUMNS attribute
  - Modified COLUMN attribute
  - Modified EXPANDABLE attribute
  - Modified FIRST–COLUMN attribute
  - Modified MOVE–COLUMN( ) method
  - Modified NEXT–COLUMN attribute
  - Modified NUM–COLUMNS attribute

  - Modified NUM–LOCKED–COLUMNS attribute
  - Modified PREVIOUS–COLUMN attribute
  - Modified ROW attribute
  - Modified X attribute
  - Modified Y attribute
  - New ALLOW–COLUMN–SEARCHING attribute

- **New supported image formats and enhancements**:
  - Modification of the DEFINE IMAGE statement
  - New STRETCH–TO–FIT attribute
  - New RETAIN–SHAPE attribute
  - New TRANSPARENT attribute
  - New IMAGE attribute
  - New IMAGE–UP attribute
  - New IMAGE–DOWN attribute
  - New IMAGE–INSENSITIVE attribute
  - New MOUSE–POINTER attribute
  - New ICON attribute
  - New SMALL–ICON attribute

- **Enhanced print capabilities**:
  - Modified PRINTER–NAME attribute for the SESSION system handle (Windows only)
  - New GET–PRINTERS( ) method for the SESSION system handle (Windows only)
  - New NUM–COPIES option for the OUTPUT TO PRINTER statement (Windows only)

- New LANDSCAPE and PORTRAIT options for the OUTPUT TO PRINTER statement (Windows only)
- New COLLATE option for the OUTPUT TO PRINTER statement (Windows only)
- New LANDSCAPE and PORTRAIT options for the SYSTEM–DIALOG PRINTER–SETUP statement (Windows only)

- **New editable combo-box widgets**:
  - Modified AUTO–ZAP attribute
  - Modified CURSOR–OFFSET attribute
  - Modified DATA–TYPE attribute
  - Modified DISABLE–AUTO–ZAP attribute
  - Modified EDIT–CAN–PASTE attribute
  - Modified EDIT–CAN–UNDO attribute
  - Modified FORMAT attribute (only for DROP–DOWN–LIST combo-box widget)
  - Modified MAX–CHARS attribute (only for SIMPLE and DROP–DOWN combo-box widgets)
  - Modified MODIFIED attribute
  - Modified SCREEN–VALUE attribute
  - Modified SELECTION–END attribute
  - Modified SELECTION–START attribute
  - Modified SELECTION–TEXT attribute
  - Modified SUBTYPE attribute
  - Modified TEXT–SELECTED attribute
  - Modified VALUE–CHANGED event
  - Modified CLEAR–SELECTION( ) method
  - Modified EDIT–CLEAR( ) method

  - Modified EDIT–COPY( ) method
  - Modified EDIT–CUT( ) method
  - Modified EDIT–PASTE( ) method
  - Modified EDIT–UNDO( ) method
  - Modified SET–SELECTION( ) method
  - Modified INNER–LINES option (only for DROP–DOWN and DROP–DOWN–LIST combo-box widgets)
  - Modified SIZE phrase
  - New AUTO–COMPLETION and UNIQUE–MATCH attributes (Windows only; Graphical interfaces only)
  - New SIMPLE (Windows only; Graphical interfaces only), DROP–DOWN (Windows only; Graphical interfaces only), and DROP–DOWN–LIST options for the COMBO–BOX phrase
  - New AUTO–COMPLETION and UNIQUE–MATCH options for the COMBO–BOX phrase (Graphical interfaces only)
  - New MAX–CHARS option for the COMBO–BOX phrase (Graphical interfaces only, and only for SIMPLE and DROP–DOWN combo-box widgets)

- **Miscellaneous widget enhancement**s:
  - Modified vertical scroll bar thumb size to reflect the relative percentage of rows displayed in a browse widget

- **Miscellaneous 4GL enhancements**:
  - Modified PUBLISH and SUBSCRIBE statements to include widget handles
  - Modified DATE function to support XML and socket input
  - Modified EXPORT statement to allow MEMPTR data type to be exported to a 4GL stream
  - Modified IMPORT statement to allow MEMPTR data type to be imported from a 4GL stream

For more information, see:

Manual:	*Progress Language Reference* and *Progress Programming Handbook*

- **Context-sensitive Help**:
  - New CONTEXT–HELP attribute
  - New CONTEXT–HELP–FILE attribute
  - New CONTEXT–HELP–ID attribute
  - Modified DEFINE VARIABLE statement
  - Modified DEFINE BUTTON statement
  - Modified DEFINE BROWSE statement
  - Modified Frame phrase

  For more information, see:

Manual:	*Progress Language Reference* and *Progress Help Development Guide*

- **Support for UNIX shared libraries**:
  - Modified implementation of the PROCEDURE statement to support declaring a routine in a UNIX shared library
  - Modified implementation of the RUN statement to support executing a routine in a UNIX shared library

  For more information, see:

Manual:	*Progress Language Reference* and *Progress External Program Interfaces*

- **Enhancements to support Internationalization:**
  - New COMPARE function
  - New COLLATE option for the FOR statement, the OPEN QUERY statement, and the PRESELECT phrase

  For more information, see:

Manual:	*Progress Language Reference* and *Progress Internationalization Guide*

1.4.5 Database

Progress Version 9.1 includes the following database enhancements:

- AutoConvert utility automatically updates prior Version 9 database metaschema to support new SQL-92 features.
- PROSTRCT UNLOCK utility provides an optional `-extent` parameter that helps access a severely damaged database.
- Binary Dump and Load utilities support binary large objects (BLOBS).

  For more information, see:

Manual:	*Progress Database Administration Guide and Reference*

1.4.6 DataServers

Version 9.1A includes enhancements to the following DataServers:

- Enterprise DataServer for ODBC
- Personal DataServer for ODBC
- DataServer for ORACLE

Progress Enterprise DataServer For ODBC

The Enterprise DataServer for ODBC has the following modifications and enhancements:

- The Enterprise DataServer for ODBC server is no longer supported on UNIX. When installed on Microsoft Windows NT, the Enterprise DataServer for ODBC server can connect to a data source residing on a UNIX server. Progress clients running on UNIX can connect to the Enterprise DataServer for ODBC server through Progress networking.
- This DataServer product now includes Version 3.5 of the MERANT ODBC drivers for all the supported ODBC data sources. For the personal DataServer for ODBC, these drivers must be purchased from the vendor. For information about these drivers, see the PDF file `installation-path\odbc\ODBCREF.PDF`.
- The Progress Explorer tool for local and remote system administration of this DataServer product has been redesigned as a snap-in to the Microsoft Management Console (MMC).
- For MS SQL Server and Sybase data sources, the ProtoODB utility allows creation of extended 4GL objects by default.
- The DataServer (–Dsrv) startup parameter has a new option, skip_schema_check, that can increase performance of your applications. Specifying this option causes the DataServer to ship schema verification of the tables the application is accessing.
- The implementation of Dynamic Queries requires recompiling of all existing applications.

  For more information, see:

Manual:	*Progress DataServer for ODBC Guide*

Progress Personal DataServer For ODBC

The Personal DataServer for ODBC has the following enhancements:

- The Progress Explorer tool for local and remote system administration of this DataServer product has been redesigned as a snap-in to the Microsoft Management Console(MMC).
- For MS SQL Server and Sybase data sources, the ProtoODB utility allows creation of extended 4GL objects by default.

- The DataServer (–Dsrv) startup parameter has a new option, skip_schema_check, that can increase performance of your applications. Specifying this option causes the DataServer to ship schema verification of the tables the application is accessing.
- The implementation of Dynamic Queries requires recompiling of all existing applications.

  For more information, see:

Manual:	*Progress DataServer for ODBC Guide*

Progress DataServer For ORACLE

The DataServer for ORACLE has the following enhancements:

- The Progress Explorer tool for local and remote system administration of this DataServer product has been redesigned as a snap-in to the Microsoft Management Console (MMC).
- In Progress Version 9.1, a single set of executables is used to access the supported versions of ORACLE (7.3 or later). Either a single client executable or a single server executable now dynamically locates and loads ORACLE shared libraries on those UNIX platforms where ORACLE supports shared objects.

  For more information, see:

Manual:	*Progress DataServer for ODBC Guide*

1.4.7 AppServer

The Progress AppServer includes the following changes:

- A new management interface. The Progress Explorer, the graphical configuration tool for local and remote system administration, has been redesigned as a snap-in to the Microsoft Management Console (MMC). The AppServer management component is also implemented as a plug-in to the Progress Explorer snap-in.

- Support for asynchronous requests from a 4GL client. In previous versions of Progress, all remote procedure calls are synchronous. That is, the 4GL client blocks until the remote procedure completes execution and returns control to the client. With new 4GL syntax, the client can call a remote procedure and continue execution without waiting for the remote procedure to return to the client. Instead, the client establishes an event procedure that executes and receives the results from the remote procedure whenever it completes execution on the AppServer.

- A new PRO*Tool, AppServer Information, allows you to view session information on a connected AppServer. For information, see the "PRO*Tools" section in this chapter.

  For more information, see:

  Manual: *Progress on the Web*

1.4.8 Open Client

The Progress Open Client Toolkit includes the following changes:

- Updated system and software requirements. These include more bundled third-party software that you do not have to install outside of the Open Client Toolkit. Also, separate Type Library files have been eliminated. ProxyGen now stores all proxy Type Library information in the single dynamic link library generated for the proxy.

- An alternate InputResultSet class that you can use in both Java 1.1 and Java 1.2 applications.

- Support for access from Java Open Clients to remote SmartDataObjects as JDBC 2 ResultSet objects. This allows you to access Progress SmartDataObjects running on the AppServer using standard JDBC methods and additional extensions to handle the unique features of SmartDataObjects.

- A new ProxyGen graphical user interface (GUI). The ProxyGen GUI is completely rewritten using the Windows Foundation Classes (WFC), providing a look and feel that is more consistent with the Windows environment.

- File association support for ProxyGen project files. This allows you to automatically start up ProxyGen by opening a project (`.pxg`) file from the Windows Explorer (or other file browser). ProxyGen then automatically opens the project associated with the opened file.

- Batch ProxyGen support on UNIX. This includes the same Batch ProxyGen support available on Windows platforms to generate Java proxies. (You cannot generate ActiveX proxies on UNIX.)
- A new utility for changing the PROPATH settings of a generated proxy project on non-Windows operating systems, like UNIX.

  For more information, see:

Manual:	*Progress Open Client Developer's Guide*

1.4.9 Internationalization

Progress Version 9.1 provides several new or revised features related to internationalization and the translation tools. The following sections summarize these features.

Translation Manager Enhancements

Version 9.1 includes the following Translation Manager enhancements:

- Expanded subset options for the File Scan, Extract, Load, Add Kit, and Resources dialog boxes
- Data Tab optimization for browse, find, and sort functions
- Enhancements to facilitate the use of third-party tools

For more information, see:

Manual:	*Progress Translation Manager Guide*

Visual Translator Enhancements

Version 9.1 includes the following Visual Translator enhancement:

- Multi-user option
- Translation tab optimization for browse, find, and sort functions

For more information, see the following documentation:

Manual:	*Progress Visual Translator Guide*

Additional Support For International Numeric Formats

Version 9.1 provides additional support for international numeric formats:

- The Fractional Separator (–numdec) startup parameter sets the character that separates the integer portion and the fractional portion of a decimal value.
- The Thousands Separator (–numsep) startup parameter sets the character that separates each group of three digits in the integer portion of a number.
- The value of NUMERIC–FORMAT can now be a character string consisting of the thousands separator followed by the fractional separator.
- The NUMERIC–SEPARATOR attribute sets the character that separates each group of three digits in the integer portion of a number.
- The NUMERIC–DECIMAL–POINT attribute sets the character that separates the integer portion and the fractional portion of a decimal value.
- The SET–NUMERIC–FORMAT() method sets the NUMERIC–SEPARATOR and NUMERIC–DECIMAL attributes to arbitrary values simultaneously.
- The –numdec parameter of Report Builder and the Report Engine sets the character that separates the integer portion and the fractional portion of a decimal value.
- The –numsep parameter of Report Builder and the Report Engine sets the character that separates each group of three digits in the integer portion of a number.
- In table dump (.d) files, the trailer contains an encoding of the numeric format of the file which has been modified in 9.1.

For more information, see:

Manual:	*Progress Startup Command and Parameter Reference*, *Progress Language Reference*, *Progress Report Builder User's Guide*, *Progress Database Administration Guide and Reference*, and Chapter 3, "Feature Comparisons," in this bulletin.

Additions To the 4GL

9.1 includes additions to the 4GL that apply to internationalization:

- The COMPARE function compares two strings, using one of several methods ("RAW," "CASE–SENSITIVE," "CASE–INSENSITIVE," or "CAPS"), and the collation table you specify.
- The COLLATE option of the FOR statement, the OPEN QUERY statement, and the PRESELECT phrase computes the collation value of a string using one of several methods ("RAW," "CASE–SENSITIVE," "CASE–INSENSITIVE," or "CAPS") and the collation table you specify.

For more information, see:

Manual:	*Progress Language Reference*

XML Code Page Support

XML documents can be encoded using any of a wide a variety of character encodings. For more information, see:

Manual:	*Progress External Program Interfaces*

SQL-92 Internationalization Support

Version 9.1 includes these new features that support SQL-92 internationalization:

- You can set the SQL_CLIENT_CHARSET_PROMSGS environment variable of an SQL-92 client to the name of a code page. If you do, the SQL-92 server converts PROMSGS run-time messages to the specified code page before sending them to the client.
- You can set the SQL_CLIENT_CHARSET_PROMSGS environment variable of an SQL-92 client to the name of a code page. If you do, the SQL-92 server converts PROMSGS run-time messages to the specified code page before sending them to the client.

  For more information, see:

Manual:	*Progress Installation and Configuration Guide Version 9 for UNIX* and *Progress Installation and Configuration Guide Version 9 for Windows*

- The SQLDUMP utility now supports the –C parameter, which lets you specify the code page of the dump file.
- The SQLLOAD utility now supports the –C parameter, which lets you specify the code page of the database utility.

For more information, see:

Manual:	*Progress Database Administration Guide and Reference*

Collations and Code-page Conversions

Version 9.1 makes a number of changes to collations and code-page conversions described in the following sections.

- **Added Collations** — Table 1–4 lists the collations added to Version 9.1.

Table 1–4: Collations Added To Version 9.1

Code Page Name	Collation Name
1250	CROATIAN
1254	TURKISH–COMP
1256	ARABIC9
1257	LITHUANIAN
620–2533	BASIC

NOTE: Progress Version 9.1 provides a new Thai collation, BASIC. This collation, available for the 620–2533 code page, follows the rules of Thai sorting more closely. The empty database in the `DLC/prolang/tai` directory uses this collation. Progress Software Corporation recommends that you use this new collation.

The collation previously named THAI is now named THAI–OLD. It is still available through the `thai-old.df` file in the `DLC/prolang/tai` directory.

For more information on converting existing Thai databases from the older collation to the newer collation, see the "Modifying Collation Tables" section of the chapter on "Character Processing Tables," in the *Progress External Program Interfaces*.

- **Renamed Collation** — Version 9.1 renames the THAI collation of code page 620–2533 to THAI–OLD.
- **Added Code-page Conversions** — Table 1–5 lists the code-page conversions added to Version 9.1.

Table 1–5: Code-page Conversions Added To Version 9.1

Source Code Page	Target Code Page
1257	UTF-8
UTF-8	1257

- **New Initial Value Segment No Convert (–isnoconv) Startup Parameter** — The –isnoconv startup parameter disables a code page conversion that Version 9.1 provides and that previous versions do not. For more information, see the following documentation:

- **Memory-mapped R-code and Automatic Code-page Conversion** — Progress Version 9.1 offers memory-mapped r-code, which reduces the amount of memory required to run multiple Progress processes. Because memory-mapped files are shared, they are read-only, which means that Progress cannot convert the code page as the file is loaded. So if the r-code code page and the internal memory code page (–cpinternal) are different, Progress reports an error and stops.

For more information, see:

Manual:	*Progress Internationalization Guide*
	Progress Database Administration Guide and Reference
	Progress Programming Handbook
	Progress Startup Command and Parameter Reference.

1.4.10 Progress SQL-92

Progress Version 9.1 includes the following enhancements to the SQL Explorer tool and to the character bit data types:

- SQL Explorer tool updates, such as new toolbar icons; changes in the Save and Save As options; and a new View menu.
- A new Options dialog box in the SQL Explorer tool that organizes properties into four categories of operation: Connection Control, Logging Control, Report Font, and Report Formatting.
- Enhanced online HTML help in the SQL Explorer tool.
- Additional SQL Explorer properties for:
  - GUI and character modes
  - Character mode only

Enhancements to the character bit data types are as follows:

- The maximum length of the VARCHAR data type has been increased.
- The bit data type has been enhanced to include LVARBINARY.

For more information, see the SQL Explorer tool online help and the following documentation:

Manual:	*Progress SQL-92 Guide and Reference,* and Chapter 3, "Feature Comparisons," in this bulletin.

- The JDBC driver in Version 9.0 has been replaced with a new industry standard JDBC driver from MERANT, Inc. formerly known as INTERSOLV. This new driver includes new JDBC class and library files. In order to use this new driver you must change your environment variables and replace references to class and library files with the new class and library filenames. For more information, see:

Manual:	*Progress JDBC Driver Guide, Progress Embedded SQL-89 Guide and Reference*

- The Progress SQL-92 ODBC driver from MERANT, Inc. formerly known as INTERSOLV, is now provided on both Windows and UNIX. For more information, see:

Manual:	*Progress DataServer for ODBC Guide, Progress Embedded SQL-89 Guide and Reference*

1.4.11 XML

Extensible Markup Language (XML) is a data format for structured document interchange over networks. This section specifies language extensions made to the Progress 4GL that enable Progress applications to use XML through the integration of a third-party DOM parser.

- The new X-document object and the following new and modified 4GL elements that support it:
  - New CREATE X–DOCUMENT statement
  - New ENCODING attribute
  - Modified NAME attribute

  - New NUM–CHILDREN attribute
  - New SUBTYPE attribute
  - Modified TYPE attribute
  - Modified UNIQUE–ID attribute
  - New APPEND–CHILD( ) method
  - New CREATE–NODE( ) method
  - New GET–CHILD( ) method
  - New GET–DOCUMENT–ELEMENT( ) method
  - New IMPORT–NODE( ) method
  - New INSERT–BEFORE( ) method
  - New LOAD( ) method
  - New REMOVE–CHILD( ) method
  - New REPLACE–CHILD( ) method
  - New SAVE( ) method

- The new X-noderef object and the following new and modified 4GL elements that support it:
  - New CREATE X–NODEREF statement
  - New ATTRIBUTE–NAMES attribute
  - New CHILD–NUM attribute
  - Modified NAME attribute
  - New NODE–VALUE attribute
  - New NUM–CHILDREN attribute
  - New OWNER–DOCUMENT attribute
  - New SUBTYPE attribute

- Modified TYPE attribute
- Modified UNIQUE–ID attribute
- New APPEND–CHILD( ) method
- New CLONE–NODE( ) method
- New DELETE–NODE( ) method
- New GET–ATTRIBUTE( ) method
- New GET–CHILD( ) method
- New GET–PARENT( ) method
- New INSERT–BEFORE( ) method
- New REMOVE–ATTRIBUTE( ) method
- New REMOVE–CHILD( ) method
- New REPLACE–CHILD( ) method
- New SET–ATTRIBUTE( ) method

For more information, see:

Manual:	*Progress Language Reference* and *Progress External Program Interfaces*

1.4.12 Sockets

Progress Version 9.1 provides 4GL support for accessing TCP/IP sockets directly rather than having to use the HLC Interface. The 4GL supports both client side and connection-oriented server side implementations. This section describes language extensions to the Progress 4GL that enable Progress applications to use TCP/IP sockets:

- New socket object and the following new and modified 4GL elements that support it:
  - New CREATE SOCKET statement
  - Modified DELETE OBJECT statement
  - New BYTES–READ attribute

- New BYTES–WRITTEN attribute
- New REMOTE–HOST attribute
- New LOCAL–HOST attribute
- Modified PRIVATE–DATA attribute
- New REMOTE–PORT attribute
- New LOCAL–PORT attribute
- Modified TYPE attribute
- Modified NEXT–SIBLING attribute
- Modified PREV–SIBLING attribute
- Modified HANDLE attribute
- Modified NAME attribute
- Modified SENSITIVE attribute
- Modified CONNECT( ) method
- New SET–READ–RESPONSE–PROCEDURE( ) method
- New READ( ) method
- New WRITE( ) method
- Modified DISCONNECT( ) method
- Modified CONNECTED( ) method
- New GET–BYTES–AVAILABLE( ) method
- New SET–SOCKET–OPTION( ) method
- New GET–SOCKET–OPTION( ) method

- New server socket object and the following new and modified 4GL elements that support it:
  - New CREATE SERVER–SOCKET statement
  - Modified DELETE OBJECT statement
  - Modified TYPE attribute
  - Modified PRIVATE–DATA attribute
  - Modified HANDLE attribute
  - Modified NAME attribute
  - Modified SENSITIVE attribute
  - New ENABLE–CONNECTIONS( ) method
  - New SET–CONNECT–PROCEDURE( ) method
  - New DISABLE–CONNECTIONS( ) method
- New socket events:
  - New READ–RESPONSE event
  - New CONNECT event
- Modified handles for socket support:
  - Modified SELF Handle
  - Modified SESSION System Handle
- Miscellaneous new and modified statements and functions for marshalling and unmarshalling data for socket support:
  - Modified LIST–EVENTS function
  - Modified LIST–QUERY–ATTRS function
  - Modified LIST–SET–ATTRS function
  - Modified LIST–WIDGETS function

  - Modified GET–DOUBLE function
  - Modified GET–FLOAT function
  - Modified GET–LONG function
  - Modified GET–SHORT function
  - Modified GET–UNSIGNED–SHORT function
  - Modified PUT–DOUBLE statement
  - Modified PUT–FLOAT statement
  - Modified PUT–LONG statement
  - Modified PUT–SHORT statement
  - Modified PUT–UNSIGNED–SHORT statement
  - Modified GET–STRING function
  - Modified PUT–STRING statement
  - New GET–BYTES function
  - New PUT–BYTES statement
- Miscellaneous new statements and functions for byte ordering and bit manipulation for socket support:
  - New SET–BYTE–ORDER statement
  - New GET–BYTE–ORDER function
  - New GET–BITS function
  - New PUT–BITS statement

For more information, see:

Manual:	*Progress Language Reference* *Progress External Program Interfaces*

1.4.13 Progress Version 9.1 Support For New Actuate APIs

In Actuate e.Reporting Suite Version 4, Actuate APIs are stored in a new Requester API DLL file called acrq4060.dll. These Actuate APIs allow Progress customers to define and invoke report-related examples from the Progress Version 9.1 application to meet their Actuate Version 4.0 report-access needs.

The following list identifies key changes and features related to Actuate Version 4.0 and Progress Version 9.1 reporting compatibility:

- New Actuate Version 4.0 and Progress Version 9.1 software component requirements to access and use the Actuate APIs.
- Numerous examples that describe how to view, generate, print, and edit parameters of Actuate reports.
- Detailed information about generic report examples that use two include files: `acapi.i` and `acpproc.i`. This information includes a description of their characteristics, how to access them using the include files, and the options available to implement these example types.
- Detailed information about specific code examples that users can modify to meet their company's reporting needs. These examples invoke the Actuate API as it is defined in the persistent procedure, `ActuateAPI.p`, and in the `ActuateAPI.i` file that contains Actuate constants.
- A list of ActiveX examples and additional Actuate report example files.

All examples referenced in the previous list are located in various Progress Version 9.1 subdirectories. For more information, see:

Manual:	*Actuate e.Reporting Suite Guide for Progress Customers, Version 5*

2

Features By Book

This chapter provides an overview of the changes to the Progress documentation set. It lists the books alphabetically in the following sections:

- Electronic Documentation
- New Documents
- Updated Documents
- Books Retired As Of This Release

2.1 Electronic Documentation

This section contains notes related to the Version 9.1 Electronic Documentation, including:

- A list of books added for the Version 9.1 Electronic Documentation
- The availability of dynamic HTML (DynaWeb), a new web-based format available for Version 9.1
- The availability of PDF, a new electronic documentation format now shipping with Version 9.1B
- The ability to search across multiple PDF files

NOTE: In addition to the new electronic documentation formats, the SGML format, viewable via DynaText is still provided on a separate CD that is shipped with your Progress product.

2.1.1 Additional Electronic Books

This section lists additional books for the Version 9.1 Electronic Documentation, available in both PDF and DynaText/DynaWeb formats:

New For Version 9.1B

- *Progress on the Web*
- *Progress DataServer for Microsoft SQL Server Guide*
- *Progress ADM 2 Guide*
- *Progress ADM 2 Reference*

New For Version 9.1A

- *Progress Client Deployment Guide*
- *Progress Startup Command and Parameter Reference*

2.1.2 New Dynamic HTML Format

As of Version 9.1, the Progress Electronic Documentation set is available on the Progress Software Corporation web site, `http://www.progress.com/v9/documentation`.

You can use any HTML browser to access the entire documentation set from any computer that is connected to the internet. This format of the documentation is provided through a server-side plug-in called DynaWeb. DynaWeb translates our SGML source books into HTML on the fly. The DynaWeb user interface and powerful search engine are similar to that of the DynaText SGML Viewer (provided with all Progress and WebSpeed products).

DynaWeb Printing

To print from DynaWeb you must have installed, on your client machine, a program or plug-in that is capable of printing PostScript files.

2.1.3 New PDF Format

In Version 9.1B, the Progress Electronic Documentation set is also available in PDF format. Each book is published as a PDF title and is available from a CD that is shipped with your Progress software as well as on the Progress Software Corporation web site:

```
http://www.progress.com/v9/documentation
```

You can view and print these PDF files using the Acrobat® Reader™, which you can download for free from Adobe® Software's Web site:

```
http://www.adobe.com/products/acrobat/readstep.html
```

For more information on using the PDF files, see the *Electronic Documentation Configuration and Installation Guide*.

2.1.4 Searching Across Multiple PDF Files

The Version 9.1C PDF files now allow you to search across multiple PDF files. For more information about this feature, see the following documentation:

Manual:	*Progress Electronic Documentation Installation and Configuration Guide*

2.2 New Documents

This section describes the new documents available with the Version 9.1 documentation set.

2.2.1 New Documents For Version 9.1B

This section lists the documents that are new for Version 9.1B:

- *Progress on the Web*

  This guide documents the following features and functionality in the Version 9.1B product:

  - Progress WebClient
  - AppServer Internet Adapter (AIA)
  - SmartObjects
  - SonicMQ Messaging

- *Progress DataServer for Microsoft SQL Server Guide*

  This guide provides instructions for building the DataServer modules, a discussion of programming considerations, a tutorial, and instructions for migrating from the DataServer for ODBC to the DataServer for Microsoft SQL Server.

- *Progress ADM 2 Guide* and *Progress ADM 2 Reference*

  In Progress Version 9.0, the *Progress AppBuilder Developer's Guide* documents both the AppBuilder application-building tool and the ADM 2 application architecture. For Version 9.1A, this manual was divided into two manuals: *Progress AppBuilder Developer's Guide* and the *Progress ADM 2 Guide and Reference*. In Version 9.1B, the *Progress ADM 2 Guide and Reference* was split into two separate manuals, the *Progress ADM 2 Guide* and *Progress ADM 2 Reference*.

  Each manual contains a core of material from the Version 9.0 manual but has been significantly expanded and revised. All the information in *Progress ADM 2 Reference* is new for Version 9.1. Each manual is described below.

  - The Version 9.1B, *Progress ADM 2 Guide* is a guide to using the Application Development Model Version 2 (ADM 2) application architecture to develop Progress application. It includes instructions for building and using Progress SmartObjects. See the guide's Preface for actual chapter descriptions.

- The Version 9.1B *Progress ADM 2 Reference* provides reference descriptions for ADM and AppBuilder functions and procedures, organized in groups by the procedure (`.p`) files that contain them. See the manual's Preface for actual chapter descriptions.

- The Version 9.1 *Progress AppBuilder Developer's Guide*, which retains the Version 9.0 title, is 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. For a in-depth description of this guide's structure, see the guide's Preface.

2.2.2 New Documents For Version 9.1A

This section lists the documents that were new for Version 9.1A:

- *Progress Client Deployment Guide*

  This guide presents an overview of the client deployment process and describes application deployment and administration tasks such as managing client access to databases, maintaining application security and user environments, managing client performance, r-code libraries, and print devices, and building Progress executables. The information in this guide was previously found in the *Progress System Administration Guide* and *Progress System Administration Reference* (both of which have been retired).

- *Progress Startup Command and Parameter Reference*

  This book provides application developers and system administrators with reference information on the Progress startup and shutdown commands and startup parameters. This information is from the *System Administration Reference*, which is now obsolete.

- *Progress Electronic Documentation Installation and Configuration Guide*

  This book describes how to install and configure Progress Electronic Documentation for Windows and UNIX platforms. It also includes information about the books and collections in the Progress Electronic Documentation Set. This book is available in hard-copy only.

- MERANT ODBC Branded Driver Reference

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

- *Progress ADM2 Guide and Reference*

  This book was derived from the *Progress AppBuilder Developer's Guide*. For Version 9.1B, this book has been split into two books: *Progress ADM 2 Guide* and *Progress ADM 2 Reference*. For more information, see the "New Documents For Version 9.1B" section.

2.3 Updated Documents

This section describes the documents in the Version 9.0 documentation set that have been updated for Version 9.1.

2.3.1 Updated Documents In Version 9.1D

- *Progress Database Administration Guide and Reference*

  This book has been updated to reflect changes made for the current release of Progress. Updates include the reordering of chapters and the addition of two new chapters: "Backup and Recovery Strategies," and "Backing Up a Database." The "The Progress Database" and "Administrative Planning" chapters have been restructured. New information can be found in the following chapters: "Creating and Deleting Databases," "After-imaging," "Dumping and Loading," "Managing Performance," "Database Startup Parameters," "Database Administration Utilities," and "Progress Monitor R&D Options."

- *Progress External Program Interfaces*

  Two new chapters and a new appendix document the Simple API for XML (SAX).

  Two chapters and an appendix on the SonicMQ Adapter were moved from the Version 9.1C *Product Update Bulletin* and updated to reflect updates for Version 9.1D.

  The chapter on sockets reflects updates for Version 9.1D.

- *Progress Installation and Configuration Guide Version 9 for Windows*

  The content of this guide has been updated throughout with the latest installation requirements for these Windows platforms: Windows NT, Windows 98, Windows XP, and CITRIX MetaFrame.

  The following list highlights the changes to this manual:

  – The installation procedures in the chapter on "Installing Progress" includes how to set up the AdminServer user-group authorization and authentication feature.

  – The chapter on "Sharing a Progress Installation on a Network" discusses the Silent Install Option users can employ for the Shared Network Installation Utility (NetSetup).

  – The chapter on "Installing Progress" has been updated to address changes to Progress messages files (`promsgs`) in the Progress Core Installation process to accommodate future Progress Add On products.

  – The chapter on "Setting Up Your Environment" includes updated information about using the AdminServer.

  – The chapter on "Configuring Progress Unified Broker Products" includes updates to address the use of the Name Server component.

  – The "Progress Version 9 Typical and Complete Installation Components" appendix has been updated with the latest list of components and subcomponents.

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

- *Progress Installation and Configuration Guide Version 9 for UNIX*

  This guide has been updated throughout with the latest installation requirements for all Version 9.1D supported UNIX platforms.

  The following list highlights the changes to this manual:

  – The chapter on "Installing Progress" provides post-installation information about setting up the AdminServer user-group authentication and authorization feature from the command line.

- The chapter on "Installing Progress" has been updated to address changes to Progress messages files (`promsgs`) in the Progress Core Installation process to accommodate future Progress Add On products.

- The chapter on "Setting Up Your Environment" provides additional details about the new parameters that support the AdminServer user-group authentication and authorization feature.

- The chapter on "Configuring Progress Unified Broker Products" includes updates to address the use of the Name Server component.

- The "Progress Version 9 Typical and Complete Installation Components" appendix has been updated with the latest list of components and subcomponents.

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

- *Progress Internationalization Guide*

  Chapter 8 has new material on the character client's limit of 31 colors when using a double-byte code page, and on the default word-break behavior of characters in double-byte code pages and UTF-8.

  Chapter 9 has new material on converting applications to UTF-8.

- *Progress Open Client Developer's Guide*

  - This guide was updated and now includes information about how the Open Client can access to an AppServer over the Internet, additional support for MEMPTR and TABLE-HANDLE parameter, and the option of connecting directly to an AppServer instead of going through a NameServer.

2.3.2 Updated Documents In Version 9.1C

This section describes the documents in Version 9.1B documentation set that were updated for Version 9.1C:

- *Pocket Progress* (hard-copy only)

  This book has been updated to reflect the changes made for the current release of Progress.

- *Progress ADM 2 Reference*

  This book has been updated with the ADM2 API reference entries from the Version 9.1B *Progress on the Web*. The book has been reorganized. New tables list the class files and properties. Individual descriptions of the property get/set routines have been eliminated, because they are so highly standardized. WebSpeed-specific entries have been gathered into one chapter.

- *Progress AppBuilder Developer's Guide*

  This book has been updated with the SmartObjects information from the Version 9.1B *Progress on the Web*. Code examples have been added for the Combo-Box widget. See the Preface for a complete description of the book's new layout.

- *Progress Application Development Environment — Getting Started*

  This guide has been updated to provide more in-depth information on the SportsPro sample application. The chapter on "Working with the Sample Application" now describes each module in the sample application and highlights important techniques. The chapter on "SmartObject Technology in the Sample Application" now lists the important objects in each module and their functions.

- *Progress DataServer for ODBC Guide*

  This guide has been updated with information on the new PRGRS_ALWAYS_INDEX option.

- *Progress Electronic Documentation Installation and Configuration Guide*

  This guide has been updated to include new information on how to search across multiple PDF files. Additionally, the DynaText installation instructions have been slightly modified.

- *Progress Installation and Configuration Guide Version 9 for Windows*

  This guide has been updated throughout with the latest installation requirements for all Windows platforms. The chapter on "Setting Up Your Environment" now covers using the AdminServer on Windows2000. The "Progress Version 9 Typical and Complete Installation Components" appendix has been updated with the latest list of components and subcomponents.

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

- *Progress Installation and Configuration Guide Version 9 for UNIX*

  This guide has been updated throughout with the latest installation requirements for all Windows platforms. The "Progress Version 9 Typical and Complete Installation Components" appendix has been updated with the latest list of components and subcomponents.

  At the request of the DG-UX vendor, the last release of Progress to be offered on the DG-UX operating system is Version 9.1B. DG-UX has been removed from the list of supported platforms for 9.1C.

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

- *Progress JDBC Driver Guide*

  This guide has been updated to refer to the latest Java requirements.

- *Progress Language Reference*

  This manual has been updated with new and modified 4GL language elements related to enhancements in the Debug Alert, the Internet Component Framework (ICF), the WebClient, and XML support. For a list of the 4GL language elements that were created or modified in Progress Version 9.1B and Version 9.1C, see Appendix D, "Progress 4GL Language Reference," in this bulletin.

- *Progress on the Web*

  This manual has been restructured to focus on developing and deploying WebClient applications. Information on SmartObjects has been moved to the *Progress AppBuilder Developer's Guide*. Information on SmartObject API has been moved to the *Progress ADM 2 Reference*. Information on the AppServer Internet Adapter and Progress SonicMQ Adapter have been moved to appendices in the *Progress Version 9 Product Update Bulletin*.

- *Progress Version 9 Product Update Bulletin*

  This bulletin has been restructured to enable you to find new information in the rest of the documentation set more easily. The quickest way to explore the new information in this release is to look at the list of cross-references in Chapter 1. You can also find large blocks of new information by reading the descriptions of new and updated books in Chapter 2. Chapter 3 highlights any changes from the way a feature worked in earlier releases to the current release. Any information on new features that has not yet been added to the document set is in the bulletin's appendices.

2.3.3 Updated Documents In Version 9.1B

This section describes the documents in Version 9.1A documentation set that were updated for Version 9.1B.

- *Progress Installation and Configuration Guide Version 9 for Windows*

  The guide has been updated throughout with the latest installation requirements for all platforms. Chapters 2, 5, 6, and 7 have been updated with information on the products added in Version 9.1B. Appendices A, C, D, and G have been updated with the latest information for this release.

- *Progress Installation and Configuration Guide Version 9 for UNIX*

  Throughout the guide, it has been updated with the latest installation requirements for all platforms. Chapters 2, 4, 5, 6, and 7 have been updated with information on the Redhat Linux platform and the products added in Version 9.1B. All the appendices have been updated with the latest information for this release.

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

  This book has been updated to include instructions on accessing and using the Progress Electronic Documentation in PDF format.

- *Pocket Progress* (hard-copy only)

  This book has been updated to reflect the changes made for the current release of Progress.

- *Building Distributed Applications Using the Progress AppServer*

  Chapter 3 has been updated to refer to the proper version of Progress.

- *Progress AppBuilder Developer's Guide*

  Appendix B has been updated with information on internationalization. Appendix C has been updated to include several new data types and a new attribute.

- *Progress Database Administration Guide and Reference*

  The syntax of the PROUTIL BUSY Qualifier has been updated. The sections on the AutoConvert utility, the Progress Backup Options, After-image Buffers parameter, and After-image I/O have been updated.

- *Progress Embedded SQL-92 Guide and Reference*

  Chapters 3, 10, and 13 have been updated to correct errors.

- *Progress External Program Interfaces*

  The DDE example has been updated.

- *Progress Internationalization Guide*

  The section on code pages has been updated.

- *Progress JDBC Driver Guide*

  Chapter 2 and Appendix A have been updated to correct some filename and path errors.

- *Progress Language Reference*

  Table 2–1 lists entries that have been updated for Version 9.1B.

Table 2–1: Progress Language Reference Updates In Version 9.1B

CHOOSE Statement	Selection-List Widget attributes list
CONNECTED Function	INDEXED-REPOSITION option for the OPEN QUERY Statement
DISABLE–TRIGGERS Statement	OS–CREATE–DIR Statement
DBTASKID Function	NO-CONSOLE option for the OS–COMMAND Statement
DYNAMIC–FUNCTION Function	NO-APPLY option for the RETURN Statement
ERROR–STATUS System Handle	SET–REPOSITIONED–ROW Method
WHERE clause for the FIND Statement and Record Phrase	STATUS Statement
FORM Statement	Status-Area attribute of the Window widget
INPUT CLEAR statement	VIEW–AS phrase
Combo-Box Widget attributes list	

- *Progress DataServer for ODBC Guide*

  Chapter 3, Chapter 4, and Appendix A have been updated to correct errors.

- *Progress Programming Handbook*

  Chapters 6, 9, and 10 have been updated for clarity and to correct errors.

- *Progress Report Builder User's Guide*

  The section on predefined parameters in Chapter 3 has been updated.

- *Progress Report Builder Tutorial*

  Parts of Chapter 3 have been updated for clarity.

- *Progress SQL-92 Guide and Reference*

  The guide has been updated to correct errors. Chapters 1, 3, and 5 have been updated with additional information.

- *Progress Startup Command and Parameter Reference*

  The description of the After-image Buffers (–aibufs) parameter has been updated.

Other Non-content Related Changes

Other changes you may notice in the Version 9.1B Electronic Documentation are:

- The presentation and formatting of certain information may have changed.
- Grammatical, consistency, stylistic, and typographical changes may have been made to some manuals.
- In the PDF files, the index sorting in some manuals may be different than in the Version 9.1A hard-copy manuals.
- In the PDF files, the page numbering may be different. Note that the page numbers referenced in the *Progress Master Glossary and Index* will **not** match the page numbers in the 9.1B PDF files.

Changed Book Titles

Table 2–2 shows the book titles that have changed in the 9.1B Electronic Documentation.

Table 2–2: Changed Book Titles

Old Title	New Title
Progress Results User's Guide for Character	*Progress Results User's Guide for UNIX*
Progress ADM 2 Guide and Reference, Volume 1	*Progress ADM 2 Guide*
Progress ADM 2 Guide and Reference, Volume 2	*Progress ADM 2 Reference*

2.3.4 Updated Documents In Version 9.1A

This section describes the documents in Version 9.0 documentation set that were updated for version 9.1A:

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

  Almost the only thing that remains unchanged from the Version 9.0 edition of this manual is the title. Material specific to ADM architecture and the construction of SmartObjects has been moved, in significantly expanded form, to the new *Progress ADM 2 Guide and Reference*.

  The remaining material has been reorganized with a view to making it more goal-oriented for the programmer. The objects that AppBuilder provides for use are now organized according to their roles in an application rather than their underlying technology. Each object type that AppBuilder supplies has been covered in more detail, with specific attention to the properties that control how the object looks and behaves.

  The extent of the reorganization would make meaningless any chapter-by-chapter comparison with the Version 9.0 edition. For information on the new organization, see the guide's Preface.

- *Progress Application Development Environment — Getting Started*

  This book was extensively reorganized. For the most part, the information remains the same but much of it now appears in different chapters. For a description of how the information is now presented, see the guide's Preface.

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

  Chapter 2, "Procedure Editor Tasks," now includes information on Procedure Editor limits taken from the *System Administration Reference*.

- *Building Distributed Applications Using the Progress AppServer*

  The following changes have been made to this book:

  - Chapter 2, "Overview of the Progress AppServer," includes an overview of asynchronous requests, server-level fault tolerance, and connection-level fault tolerance.

  - Chapter 3, "Administration," includes information on using the new MMC-based Progress Explorer tool to manage AppServers. It is also no longer the central location for information on the NameServer or the distribution of machine resources. This information has been collected and consolidated within a separate chapter on Progress Unified Broker products, versions of which exist in both the *Progress Installation and Configuration Guide Version 9 for Windows* and the *Progress Installation and Configuration Guide Version 9 for UNIX*. Otherwise, the chapter has been reorganized and rewritten to accommodate the specific focus on AppServer components and includes reference information on the command-line utilities that previously resided in the Version 9.0 *Progress System Administration Reference*.

  - Chapter 5, "Programming the Client Application," is extensively revised to include information on programming asynchronous AppServer requests.

  - Chapter 6, "Design and Implementation Considerations" (changed from "Design and Implementation"), has a new section on the asynchronous request queuing model and revised information on using AppServer operating modes.

  - Chapter 7, "Debugging," has a small update explaining how to debug asynchronous requests.

- *Progress DataServer for ODBC Guide*

  All information pertaining to the operation of the DataServer server on UNIX has been removed from this book.

- *Progress DataServer for ORACLE Guide*

  The following changes have been made to this book:

  - Chapter 5, "The DataServer Tutorial," now describes the Adjust Schema utility for migrating changes from an ORACLE database.
  - Appendix A, "Upgrading DataServer Applications," a section on manually modifying your ORACLE tables, previously deleted from the manual, has been restored.
  - Appendix E, "Building DataServer Executables," has a section on building executables on the UNIX client that has been updated to better describe the current requirements.

- *Progress Database Administration Guide and Reference*

  The following changes have been made to this book:

  - The chapter "Managing User Licences" has been removed. The information is now presented in the *Progress Installation and Configuration Guide Version 9 for UNIX*, and the *Progress Installation and Configuration Guide Version 9 for Windows*.

- *Progress Debugger Guide*

  The following changes have been made to this book:

  - Chapter 7, "DEBUGGER and SESSION System Handles," now includes a section that documents how to use an alert box to access a Stack Trace dialog box. The Stack Trace dialog box displays 4GL stack trace information for the currently executing Progress session.

- *Progress Developer's Toolkit*

  The *Progress Developer's Toolkit* documentation now includes conceptual diagrams and information at the beginning of Chapter 2, "Choosing a Code Format," and Chapter 3, "Initial Deployment." This information is intended to help users choose the deployment strategy that is most appropriate for their applications. The documentation has also been updated to reflect the current version of the Toolkit.

- *Progress External Program Interfaces*

  The following changes have been made to this book:

  - Chapter 1, "Introduction," now documents the MEMPTR data type for all EPIs that reference it and adds introductory sections for the latest EPI support.
  - Chapter 4, "UNIX Named Pipes," has been renamed to "Named Pipes." Information on Windows NT named pipes has been added.
  - Chapter 5, "Windows Dynamic Link Libraries," has had most information on the MEMPTR data type moved to Chapter 1. Information specific to DLLs has been retained.
  - Chapter 10, "Sockets," is new and describes how to use socket objects for data transmission and 4GL processing.
  - Chapter 11, "XML Support," is new and describes how to use XML objects for data transmission and 4GL processing.
  - All sample programs have been updated to 32-bit DLLs where necessary.

- *Progress Help Development Guide* (Windows only)

  The following changes have been made to this guide:

  - Chapter 1, "Providing Help for Progress Applications," describes how to implement context-sensitive help with the Progress 4GL and with the AppBuilder.
  - Chapter 3, "Creating Help Source Files," no longer contains the table, "Help Macro Summary." This information is in the *Help Author's Guide*, a Microsoft help system that Progress provides to accompany the Microsoft Help compiler.
  - Appendix A, "Files Mentioned in This Guide," is new and lists the name, location, and description of each file mentioned in this guide.
  - Appendix B, "Options in the [Options] Section of the Project File," is a long table formerly in Chapter 4, "Completing Help Projects."

- *Progress Installation and Configuration Guide Version 9 for UNIX*

  For Version 9.1, the installation guide's title changed from *Progress Installation Notes Version 9 for UNIX* to *Progress Installation and Configuration Guide (UNIX)*. This guide has been reorganized and expanded, and now contains information on both installing and configuring Progress. For an in-depth description of the new structure, see the guide's Preface.

  The following sections have been added to this book from the *Progress System Administration Guide* Version 9.0:

  - The material from Chapter 3, "Configuring Progress," of the *Progress System Administration Guide* Version 9.0 is now included in Chapter 7, "Configuring Progress."
  - The material on starting Progress and running Progress clients and servers on a network from Chapter 5, "Starting Up and Shutting Down," of the *Progress System Administration Guide* Version 9.0 is now included in Chapter 7, "Configuring Progress."
  - The material from Chapter 8, "Managing User Licenses," of the *Progress System Administration Guide* Version 9.0 is now included in Chapter 3, "System Administration Utilities."
  - The material from Chapter 19, "Progress Event Logging," of the *Progress System Administration Guide* Version 9.0 is now included in Chapter 3, "System Administration Utilities."

  The following sections have been added to this book from the *Progress System Administration Reference* Version 9.0:

  - The material from Chapter 2, "Environment Variables," of the *Progress System Administration Reference* Version 9.0 is now included in Chapter 5, "Setting Up Your Environment."
  - The material on using the AdminServer from Chapter 2, "Administration Utilities," of the *Progress System Administration Reference* Version 9.0 is now included in Chapter 5, "Setting Up Your Environment."

- The material on using the NSCONFIG utility from Chapter 5, “Administration Utilities,” of the *Progress System Administration Reference* Version 9.0 is now included in Chapter 3, “System Administration Utilities.“

- The material on using the SHOWCFG utility from Chapter 5, “Administration Utilities,” of the *Progress System Administration Reference* Version 9.0 is now included in Chapter 3, “System Administration Utilities.”

- *Progress Installation and Configuration Guide Version 9 for Windows*

  For Version 9.1, the installation book title changed from *Progress Installation Notes Version 9 for Windows* to *Progress Installation and Configuration Guide (Windows)*. This book has been reorganized and expanded, and now contains information on both installing and configuring Progress. For an in-depth description of the new structure, see the guide’s Preface.

  The following sections have been added to this book from the *Progress System Administration Guide* Version 9.0:

  - The material from Chapter 3, “Configuring Progress,” of the *Progress System Administration Guide* Version 9.0 is now included in Chapter 6, “Configuring Progress.”

  - The material on starting Progress and running Progress clients and servers on a network from Chapter 5, “Starting Up and Shutting Down,” of the *Progress System Administration Guide* Version 9.0 is now included in Chapter 6, “Configuring Progress.”

  - The material from Chapter 8, “Managing User Licenses,” of the *Progress System Administration Guide* Version 9.0 is now included in Chapter 4, “Administration Utilities.”

  - The material from Chapter 19, “Progress Event Logging,” of the *Progress System Administration Guide* Version 9.0 is now included in Chapter 4, “Administration Utilities.”

  The following sections have been added to this book from the *Progress System Administration Reference* Version 9.0:

  - The material from Chapter 2, “Environment Variables,” of the *Progress System Administration Reference* Version 9.0 is now included in Chapter 5, “Setting Up Your Environment.”

- The material on using the AdminServer from Chapter 2, “Administration Utilities,” of the *Progress System Administration Reference* Version 9.0 is now included in Chapter 5, “Setting Up Your Environment.”
- The material on using the NSCONFIG utility from Chapter 5, “Administration Utilities,” of the *Progress System Administration Reference* Version 9.0 is now included in Chapter 6, “Configuring Progress.”
- The material on using the ProControl utility from Chapter 5, “Administration Utilities,” of the *Progress System Administration Reference* Version 9.0 is now in Appendix B, “The ProControl Utility.”
- The material on using the Progress Explorer utility from Chapter 5, “Administration Utilities,” of the *Progress System Administration Reference* Version 9.0 is now included in Chapter 6, “Configuring Progress.”
- The material on using the SHOWCFG utility from Chapter 5, “Administration Utilities,” of the *Progress System Administration Reference* Version 9.0 is now included in Chapter 4, “Administration Utilities.”

- *Progress Internationalization Guide*

  This book has been completely reorganized and rewritten for Version 9.1. For an in-depth description of the new structure, see the guide’s Preface.

- *Progress JDBC Driver Guide*

  This book has been completely revised to describe the new JDBC driver from MERANT.

- *Progress Language Reference*

  Table 2–3 lists attributes that have been added to this manual.

Table 2–3: New Attributes *(1 of 2)*

ALLOW-COLUMN-SEARCHING	IMAGE-UP
ASYNC-REQUEST-COUNT	LOCAL-HOST
ATTRIBUTE-NAMES	LOCAL-PORT
AUTO-COMPLETION	MOUSE-POINTER
AUTO-VALIDATE	NODE-VALUES

Table 2–3: New Attributes *(2 of 2)*

BUFFER-FIELD	NUM-CHILDREN
BYTES-READ	NUM-VISIBLE-COLUMN
BYTES-WRITTEN	OWNER-DOCUMENT
CANCELLED	PERSISTENT-PROCEDURE
CHILD-NUM	PREPARED
COMPLETE	PRIMARY
CONTEXT-HELP	PROCEDURE-NAME
CONTEXT-HELP-FILE	QUIT
CONTECXT-HELP-ID	REMOTE-HOST
DEFAULT-BUFFER-HANDLE	REMOTE-SHAPE
ENCODING	RETAIN-SHAPE
ERROR	SMALL-ICON
EVENT-PROCEDURE	STOP
EVENT-PROCEDURE-CONTEXT	STRETCH-TO-FIT
FIRST-BUFFER	SUBTYPE
ICON	TABLE-HANDLE
IMAGE	TRANSPARENT
IMAGE-DOWN	UNDO
IMAGE-SENSITIVE	UNIQUE-MATCH

Table 2–4 lists events that have been added to this manual.

Table 2–4: New Events

CONNECT	READ–RESPONSE
PROCEDURE–COMPLETE	

Table 2–5 lists functions that have been added to this manual.

Table 2–5: New Functions

COMPARE	GET–BYTE–ORDER
GET–BITS	GET–BYTES

Table 2–6 lists object handles that have been added to this manual.

Table 2–6: New Object Handles

Asynchronous request	X–DOCUMENT
SERVER–SOCKET	X–NODEREF
SOCKET	

Table 2–7 lists methods that have been added to this manual.

Table 2–7: New Methods *(1 of 2)*

ADD–CALC–COLUMN( )	GET–BYTES–AVAILABLE( )
ADD–COLUMNS–FROM( )	GET–CHILD( )
ADD–FIELDS–FROM( )	GET–DOCUMENT–ELEMENT( )
ADD–INDEX–FIELD( )	GET–PARENT( )
ADD–LIKE–COLUMN( )	GET–PRINTERS( )
ADD–LIKE–FIELD( )	GET–SOCKET–OPTION( )

Table 2–7: New Methods *(2 of 2)*

ADD–LIKE–INDEX( )	IMPORT–NODE( )
ADD–NEW–FIELD( )	INDEX–INFORMATION( )
ADD–NEW–INDEX( )	INSERT–BEFORE( )
APPEND–CHILD( )	LAST–ASYNC–REQUEST( )
BUFFER–COMPARE( )	LOAD( )
BUFFER–COPY( )	RAW–TRANSFER( )
CANCEL–REQUESTS( )	READ( )
CLEAR( )	REMOVE–ATTRIBUTE( )
CLONE–NODE( )	REMOVE–CHILD( )
CREATE–LIKE( )	REPLACE–CHILD( )
CREATE–NODE( )	SAVE( )
DELETE–NODE( )	SET–ATTRIBUTE( )
DISABLE–CONNECTIONS( )	SET–CONNECT–PROCEDURE( )
ENABLE–CONNECTIONS( )	SET–READ–RESPONSE–PROCEDURE( )
FIRST–ASYNC–REQUEST( )	SET–SOCKET–OPTION( )
GET–ATTRIBUTE( )	TEMP–TABLE–PREPARE( )
GET–BROWSE–COLUMN( )	WRITE( )

Table 2–8 lists options that have been added to this manual.

Table 2–8: New Options

Option	Applies To . . .
ASYNCHRONOUS	RUN statement
AUTO–COMPLETION	COMBO–BOX phrase
COLLATE	OUTPUT TO PRINTER statement
DROP–DOWN	COMBO–BOX phrase
LANDSCAPE	OUTPUT TO PRINTER and SYSTEM–DIALOG PRINTER–SETUP statements
MAX–CHARS	COMBO–BOX phrase
NUM–COPIES	OUTPUT TO PRINTER statement
PORTRAIT	OUTPUT TO PRINTER and SYSTEM–DIALOG PRINTER–SETUP statements
SIMPLE	COMBO–BOX phrase
UNIQUE–MATCH	COMBO–BOX phrase

Table 2–9 lists statements that have been added to this manual.

Table 2–9: New Statements

CREATE BROWSE	CREATE X–NODEREF
CREATE SERVER–SOCKET	PUT–BITS
CREATE SOCKET	PUT–BYTES
CREATE TEMP–TABLE	SET–BYTE–ORDER
CREATE X–DOCUMENT	

Table 2–10 lists attributes that have been modified in this release.

Table 2–10: Modified Attributes

AUTO–ZAP	PREVIOUS–COLUMN
COLUMN	PREV–SIBLING
CURSOR–OFFSET	PRINTER–NAME
DATA–TYPE	PRIVATE–DATA
DISABLE–AUTO–ZAP	QUERY
DYNAMIC	ROW
EDIT–CAN–PASTE	SCREEN–VALUE
EDIT–CAN–UNDO	SELECTION–END
EXPANDABLE	SELECTION–START
FIRST–COLUMN	SELECTION–TEXT
FORMAT	SENSITIVE
HANDLE	SERVER
LABELS	SUBTYPE
MAX–CHARS	TEXT–SELECTED
MODIFIED	TYPE
MULTIPLE	UNIQUE–ID
NAME	VALIDATE–EXPRESSION
NEXT–COLUMN	VISIBLE
NEXT–SIBLING	X
NUM–COLUMNS	Y
NUM–LOCKED–COLUMNS	

Table 2–11 lists the event that has been modified in this release.

Table 2–11: Modified Events

VALUE–CHANGED

Table 2–12 lists functions that have been modified in this release.

Table 2–12: Modified Functions

DATE	GET–UNSIGNED–SHORT
GET–DOUBLE	LIST–EVENTS
GET–FLOAT	LIST–QUERY–ATTRS
GET–LONG	LIST–SET–ATTRS
GET–SHORT	LIST–WIDGETS
GET–STRING	

Table 2–13 lists handles that have been modified in this release.

Table 2–13: Modified Handles

SELF system handle	SESSION system handle
Server object handle	

Table 2–14 lists methods that have been modified in this release.

Table 2–14: Modified Methods

ADD–SUPER–PROCEDURE( )	EDIT–COPY( )
CLEAR–SELECTION( )	EDIT–CUT( )
CONNECT( )	EDIT–PASTE( )
CONNECTED( )	EDIT–UNDO( )
DISCONNECT( )	MOVE–COLUMN( )
EDIT–CLEAR( )	SET–SELECTION( )

Table 2–15 lists options that have been modified in this release.

Table 2–15: Modified Options

INNER–LINES	DROP–DOWN–LIST

Table 2–16 lists phrases that have been modified in this release.

Table 2–16: Modified Phrases

COMBO–BOX	SIZE
Frame	

Table 2–17 lists statements that have been modified in this release.

Table 2–17: Modified Statements *(1 of 2)*

CREATE BUFFER	PUT–DOUBLE
DEFINE BROWSE	PUT–FLOAT
DEFINE BUTTON	PUT–LONG
DEFINE IMAGE	PUT–SHORT

Table 2–17: Modified Statements *(2 of 2)*

DEFINE PARAMETER	PUT–STRING
DEFINE VARIABLE	PUT–UNSIGNED–SHORT
DELETE OBJECT	RUN
EXPORT	SUBSCRIBE
IMPORT	SYSTEM–DIALOG PRINTER–SETUP
PROCEDURE	OUTPUT TO PRINTER
PUBLISH	WAIT–FOR

- *Progress DataServer for ODBC Guide*

  The following change has been made to this book:

  – Chapter 5, “The UNIX Environment,” has been added. This chapter contains specific information about using the Progress SQL-92 ODBC Driver in the UNIX environment.

- *Progress Open Client Developer’s Guide*

  The following changes have been made to this book:

  – Chapter 1, “Overview,” contains new information on accessing remote SmartDataObjects.

  – Chapter 2, “Configuration and Deployment” (changed from “System Requirements”), has been revised to cover recent Java developments and the new architecture for handling Type Library information for ActiveX proxies.

  – Chapter 3, “Generating and Managing Proxies,” is updated to reflect the new WFC-based interface and functional changes in both GUI and Batch ProxyGen. It also includes information on the new ChangePropath utility.

  – Chapter 4, “Programming Concepts,” contains new information on accessing remote SmartDataObjects from a Java Open Client.

  – Chapter 5, “Programming Java Clients,” contains new information on alternate InputResultSet classes and accessing remote SmartDataObjects.

- Chapter 6, “Using SmartDataObjects from Java Clients,” is new. It describes all the new classes for accessing remote SmartDataObjects and how to use them.
- Chapter 7 (previously Chapter 6), “Programming ActiveX Controller Clients,” clarifies the use of early and late binding, Progress data type requirements, how to access fields in TEMP–TABLE parameters, and using proxy runtime properties.
- Appendix A, “Accessing a SmartDataObject API Directly” (changed from “Accessing SmartDataObjects”), focuses on what to do if you must access a remote SmartDataObject as a ProcObject and why you might need to do it. The content has been greatly reduced in order to reference the more complete treatment of the SmartDataObject API provided by the new *Progress ADM 2 Guide and Reference*.

- *Pocket Progress*

  This book has been updated to reflect the changes made for the current release of Progress.

- *Progress Programming Handbook*

  The following changes have been made to this book:

  - Chapter 3, “Block Properties,” has been updated with information on the super procedure search rules and a new parameter of the ADD–SUPER PROCEDURE method that allows you to change the default search when a super procedure executes a RUN SUPER statement.
  - Chapter 4, “Named Events,” has been updated with information on publishing events from widgets.
  - Chapter 7, “Alternate I/O Sources,” has new sections, “Getting a List of Printers” and “Changing the Default Printer,” that describe how to use the GET–PRINTERS and PRINTER–NAME options. It also adds syntax information on the Quoter utility, a new table on printing options for Windows
  - Chapter 9, “Database Access,” has had the “Query, Buffer, and Buffer-field Objects” section removed and transferred to Chapter 20, “Using Dynamic Widgets.” In addition, the section on word indexing has been completely reorganized and rewritten.
  - Chapter 10, “Using the Browse Widget,” has been updated with new information on the INSERT–ROW() method and the removal of the section “Creating Browse Rows in an Empty Browser,” which describes a procedure that is no longer necessary.

- Chapter 15, “Work Tables and Temporary Tables” has a new section, “Dynamic Temporary Tables,” that describes creating and using temp-table objects and changes to the static temp-table.
- Chapter 18, “Buttons, Images, and Rectangles,” has been updated with new information on images.
- Chapter 20, “Using Dynamic Widgets,” has a new section, “Creating and Using Dynamic Queries,” which is an expanded version of the old “Query, Buffer, and Buffer-field Objects” section from Chapter 9, “Database Access.”
- Chapter 20, “Using Dynamic Widgets,” has a new section, “Creating and Using a Dynamic Browse,” which describes creating and using a dynamic browse and changes to the static browse.
- Appendix A, “R-code Features and Functions,” has been updated with information about standard and memory-mapped r-code libraries.

- *Progress SQL-92 Guide and Reference*

  This book has been updated to describe enhanced and new data types. The book includes new information on the enhanced data types VARCHAR and VARBINARY, and on the new data type LVARBINARY.

- *Progress Translation Manager Guide*

  The following changes have been made to this book:

  - Chapter 1, “Preparing Your Application for Translation,” describes issues developers should be aware of and techniques they should use to facilitate translation, and includes information on designing an interface for localization formerly found in Appendix B, “Developing Translatable Applications.”
  - Chapter 5, “Preparing Data for Translation,” includes information on the expanded options now available with the following processes: file scan, extract, load, resources, and add kit. Once you have completed these processes and created a kit for the first time, you can choose to work only on new and changed procedures in subsequent iterations of the project.
  - Chapter 10, “Tutorial,” guides you through a sample translation project.
  - Appendix B, “Developing Translatable Applications,” is no longer included in this book. Its contents have been incorporated into Chapter 1.

- *Progress Visual Translator Guide*

  The following changes have been made to this book:

  - Chapter 2, "Getting Started," includes information on working on a translation project in multi-user mode, which allows multiple translators to work on the same translation project using the same glossary.
  - Appendix A, "Installing Visual Translator," includes instructions for installing Visual Translator.
  - Appendix B, "Visual Translator Interface Reference," contains information formerly found in Appendix A.

- *Welcome to Progress*

  This book has been updated to reflect the current release of Progress, and to include current corporate information.

2.4 Books Retired As Of This Release

This section describes the documents in the Version 9.0 documentation set that have been retired for Version 9.1:

- *Progress/400 Installation Notes*
- *Progress System Administration Guide*
- *Progress System Administration Reference*

3

Feature Comparisons

This chapter provides comparison about how the behavior of features in this version differs in previous Version 9 products. Descriptions of new features not documented in another Progress manual are in the appendices of the this bulletin. This chapter includes information on:

- Administration
- Bit String Data Type Enhancements
- Databases
- DataServers
- Electronic Documentation
- Installation
- Internationalization
- OCX Events
- Progress Explorer Tool
- Progress 4GL Enhancements
- Progress SQL-92
- SmartObjects
- WebClient Enhancements

3.1 Administration

Please note the following changes in feature behavior:

- AdminServer Security
- Debug Alert (-debugalert) Startup Parameter
- R-code Action Code Segments

3.1.1 AdminServer Security

Progress Version 9.1D has enhanced functionality that provides administrators with the opportunity to establish more security when it comes to working with the AdminServer. A feature of the AdminServer security enhancement is that a username and password can now be required for starting the AppServer, Sonic Adapter, and WebSpeed. The difference in behavior in Progress Version 9.1D and prior versions regarding the require username option for starting the AppServer, WebSpeed, and Sonic Adapter is documented in the following sections:

- Require Username and Password On Windows
- Require Username and Password On Unix

Require Username and Password On Windows

On the Windows platform in versions earlier than Progress Version 9.1D, the username is optional to start the servers, but if you provide a username, you also have to provide a password. Progress verifies that the username and password are valid through authentication. In Progress Version 9.1D, the username and password are still optional by default, but you can now require a username and password using the Progress Explorer or the command-line option Require Username (`-requireusername`). Additionally, blank passwords are now supported.

For additional information on how to require username and password, see:

Manual:	Appendix F, “Administration” in this bulletin.

Require Username and Password On Unix

On the UNIX platform in versions earlier than Progress Version 9.1D, the username is optional, the password is never required, and Progress never authenticates the starting of the servers. In Progress Version 9.1D, the default username is still optional, but if you supply a username, a password is now required. Additionally, you can require a username and password using the Progress Explorer or with the command-line option Require Username (`-requireusername`). If you require the username and password, Progress Version 9.1D on UNIX now verifies that the username and password are valid through authentication. Additionally, blank passwords are now supported.

For additional information on how to require username and password, see:

Manual:	Appendix F, "Administration" in this bulletin.

3.2 Debug Alert (-debugalert) Startup Parameter

Prior to Progress Version 9.1C, the Debug Alert (`-debugalert`) startup parameter provided access to 4GL stack trace information when an error occurred during an interactive session. Progress displayed an alert box with a help button that displayed a dialog box containing a scrollable, read-only 4GL stack trace for the current Progress session.

In Progress Version 9.1C, the Debug Alert (`-debugalert`) startup parameter also provides access to 4GL stack trace information when an error occurs during a batch session. Progress records the 4GL stack trace information in the `procore` file.

In Progress Version 9.1D, the Debug Alert (`-debugalert`) startup parameter, when used with the new `-clientlog` startup parameter, provides access to 4GL stack trace information during a session and writes this information to a log file you specify. In version 9.1C, the information was written to the `procore` file, in Version 9.1D, information is written to the log file you specify with the `-clientlog` startup parameter. If you do not use `-clientlog` to specify a log file name, messages are not written to a log file.

For more information about Debug Alert, see:

Manual:	Appendix D, "Progress 4GL Language Reference" in this bulletin. and *Progress Startup Command and Parameter Reference*.

3.3 R-code Action Code Segments

R-code is the intermediate binary code that Progress generates when it compiles 4GL source files. A Progress r-code file is divided into segments of varying type. One of these segment types is an action code segment. An action code segment holds executable code in the form of action cells that drive the Progress interpreter and contain the executable code for 4GL verbs.

Prior to Progress Version 9.1C, an r-code file contained one action code segment for the main procedure, and one separate action code segment for each internal procedure.

Progress Version 9.1C now supports up to four main action code segments for the main procedure, and continues to support one separate action code segment for each internal procedure.

For more information about r-code and its structure, see:

Manual:	Appendix B, "R-code Features and Functions," in this bulletin.

3.4 Bit String Data Type Enhancements

This section describes the changes to the VARBINARY and VARCHAR data types, and the new LONG VARBINARY data type.

This is the new syntax for a `bit_string_data_type`:

SYNTAX

```
BIT | BINARY | VARBINARY | LVARBINARY [ ( length ) ]
```

`VARBINARY ( length )`

Type `VARBINARY` corresponds to a variable-length bit field of the specified length. The maximum length is 31995.

`LVARBINARY ( length )`

Type `LVARBINARY` corresponds to an arbitrarily long byte array with the maximum length defined by the amount of available disk storage up to 2,000,000,000. A BLOB is an object of data type `LVARBINARY`.

3.4.1 Maximum Length For VARBINARY

Specifically, the maximum length of the VARBINARY data type is:

- 31995 bytes for single-column tables converted from Version 8 to Version 9 and not updated in Version 9.
- 31982 bytes for single-column tables originally created in Version 9 or updated in Version 9.

The maximum length of the VARBINARY data type depends on:

- The number of columns in a table — More columns in a table further limits the length of VARBINARY data.
- When a table was created — Tables created earlier can support longer VARBINARY data than tables created later.

3.4.2 Character Data Type Enhancements

This section describes the changes to the data type VARCHAR.

This is the new syntax for a `char_data_type`:

SYNTAX

```
{ CHARACTER | CHAR } [ ( length ) ]
  | { CHARACTER VARYING | CHAR VARYING | VARCHAR } [ ( length ) ]
```

`VARCHAR [ ( length ) ]`

Type `VARCHAR` corresponds to variable-length character string with the maximum length specified. The default length is 1. The maximum length is 31995.

Maximum Length For VARCHAR

Specifically, the maximum length or maximum number of characters for the VARCHAR data type is:

- 31995 characters for single-column tables converted from Version 8 to Version 9 and not updated in Version 9.
- 31982 characters for single-column tables originally created in Version 9 or updated in Version 9.

The maximum length of the VARCHAR data type depends on:

- The number of columns in a table — More columns in a table further limits the length of VARCHAR data.
- When a table was created — Tables created earlier can support longer VARCHAR data than tables created later.

National Language Support (NLS)

The VARCHAR data type has NLS. The choice of character set affects the available character count or maximum length of the data column. The limits established above assume a single-byte character set. Using a multiple-byte character set lowers the maximum character count proportionally. For example, if all the characters in a character set take 3 bytes per character, in Version 9 the practical maximum is 10660 (31982 divided by 3).

3.5 Databases

Please note the following change in feature behavior: self-tuning `-pwqdelay` parameter.

3.5.1 Self-tuning -pwqdelay Parameter

In Version 9.1D, `-pwqdelay` is self-tuning:

- **Old Behavior —** The user adjusted `-pwqdelay` to determine how many milliseconds elapsed before the Asynchronous Page Writer (APW) checked the APW queue.
- **New Behavior —** The user can still set the value of `-pwqdelay`, but it is a self-tuning parameter. When the database encounters heavy loads, `-pwqdelay` decreases its value so that the APW writes more often. When the demand on the database lessens, `-pwqdelay` increases its value so that the APW writes less often.

3.6 DataServers

Please note the following changes in feature behavior:

- Single Executable for DataServer for ORACLE on UNIX

3.6.1 Single Executable For DataServer For ORACLE On UNIX

In earlier versions of Progress, separate sets of executables (both client and sever) were required to access ORACLE data sources, depending on the ORACLE version (7.x or 8.x). In Progress Version 9.1 a single set of executables is used to access the supported versions of ORACLE (7.3 or later). Either a single client executable or a single server executable now dynamically locates and loads ORACLE shared libraries on those UNIX platforms where ORACLE supports shared libraries.

The single executable dynamically locates and loads the ORACLE shared libraries. This means there is no need to set the LD_LIBRARY_PATH environment variable, and there is no need to create a softlink in `/usr/lib` that points to the shared objects. You must, however, specify the environment variable ORACLE_HOME by giving the pathname of the directory on your system where you installed ORACLE.

Table 3–1 lists descriptions of the client and host DataServer executables for Version 9.1.

Table 3–1: UNIX DataServer For ORACLE Executables

Executable	Description
_orarx	Client executable that includes the local DataServer module. Use this executable for a local DataServer configuration or a configuration that accesses ORACLE Version 7.3 and later through SQL*Net or Net 8.
_orasrv	Remote DataServer executable that resides on the host machine. This is the executable that `probrkr` calls. Use it for a remote DataServer configuration that accesses ORACLE Version 7.3 and later.

3.7 Electronic Documentation

Please note the following change in feature behavior:

- PDF Search Capabilities in Version 9.1C

3.7.1 PDF Search Capabilities In Version 9.1C

In Version 9.1C, you can now search across multiple PDF files.

- **Old behavior** — In Version 9.1B, you could only perform simple "Find" function for a single PDF file.
- **New behavior** — In Version 9.1C, you can now perform an advanced "Search" function which allows you to search across multiple PDF files at one time.

For information on how to use the new PDF search functionality, see

Manual:	The *Progress Electronic Documentation Installation and Configuration Guide*.

3.8 Installation

This section provides information about changes to the Progress Core installation.

3.8.1 Changes In the Progress Core Installation Process For Progress Message Files

During the Progress Core Installation process prior to Progress Version 9.1D, there is a potential to incorrectly overwrite or update a `promsgs` file in the local directory. When re-installing Progress, as is necessary to perform various tasks, there is no mechanism to ensure that the most up-to-date promsgs files are always available. Prior to Progress Version 9.1D, the patch process has been the vehicle to address file changes and updates, including addressing incorrectly overwritten `promsgs` files, and files such as `messages.jar` and Prohelp\`msgdata` that are associated with `promsgs` files.

Progress Version 9.1D incorporates the following changes to address this problem:

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

In Progress Version 9.1D, the Progress Core Installation program default behavior determines that if a `promsgs` file exists in the local directory, it is the latest version. Therefore, no file changes will be performed to any `promsgs` files in this situation.

- Specific file protection measures have also been added to accommodate files associated with Progress Add On products. Progress Add On Products refer to Progress products released independently of a point or major Progress product release. However, these products work with and depend on functionality available within the Progress Core software product set.

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

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

3.9 Internationalization

Please note the following changes in feature behavior:

- Modified Encoding of the Numeric Format in the Trailer of Table Dump (.d) Files
- Clarifications Concerning the Decimal Point and the Thousands Separator
- THAI Collation of Code Page 620–2533 Renamed
- New Initial Value Segment No Convert (`-isnoconv`) Startup Parameter

3.9.1 Modified Encoding Of the Numeric Format In the Trailer Of Table Dump (.d) Files

In table dump (`.d`) files, the trailer contains an encoding of the numeric format of the file.

In Progress prior to Version 9.1, the format of the encoding is:

SYNTAX

```
numformat=comma-or-period
```

`comma-or-period`

A comma if the numeric format follows the European standard (in which the decimal point is a comma and the thousands separator is a period) or a period if the numeric format follows the American standard (in which the decimal point is a period and the thousands separator is a comma).

In Progress Version 9.1, the format of the encoding is:

SYNTAX

```
numformat=thousands-separator,decimal-point
```

`thousands-separator`

The numeric value of the character that represents the thousands separator in the table dump file, using the table dump file's code page.

`decimal-point`

The numeric value of the character that represents the decimal point in the table dump file, using the table dump file's code page.

In Version 9.1, Progress reports a runtime error if one of the following occurs:

- SESSION:NUMERIC–FORMAT = "EUROPEAN" and the table dump file trailer contains the line "numformat=. " which indicates the American standard.
- SESSION:NUMERIC–FORMAT = "AMERICAN" and the table dump file trailer contains the line "numformat=," which indicates the European standard.

3.9.2 Clarifications Concerning the Decimal Point and the Thousands Separator

This section clarifies the changes described in the previous section.

Numeric Constants In 4GL Source Code

In 4GL source code, in numeric constants, Progress still expects the fractional separator to appear as a period and the thousands separator to appear as a comma. This has not changed.

Word-break Tables and Word Indexing

If you change the character that represents, in formatted text, the fractional separator or the thousands separator, and you want word breaking to conform to these changes, you must set up your word-break table accordingly. For more information see the word-break tables and word indexing, in the:

Manual:	*Progress Programming Handbook* and the *Progress Internationalization Guide*

3.9.3 THAI Collation Of Code Page 620–2533 Renamed

Version 9.1 renames the THAI collation of code page 620–2533 to THAI–OLD. You must change any existing references to THAI in your applications to THAI–OLD.

For more information on the collations and code-page conversions provided by Version 9.1, see the README file located in the `DLC/prolang` directory.

3.9.4 New Initial Value Segment No Convert (–isnoconv) Startup Parameter

The `–isnoconv` startup parameter disables a code page conversion that Version 9.1 provides and that previous versions do not. You might consider using `–isnoconv` when all three of the following conditions occur:

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

When you run earlier versions of Progress under these conditions, when Progress reads text literals from r-code into internal memory, Progress does not convert the code page. To compensate for this, programmers often preconvert international characters in text literals to `-cpinternal`.

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

- International characters preconverted (by the programmer) to `-cpinternal` are converted again (by Progress) to `-cpinternal`, garbling them.
- The text segment might grow beyond its maximum size.

To disable this code-page conversion in Progress Version 9.1, use the `-isnoconv` startup parameter.

3.10 OCX Events

The handling of asynchronous COM events has changed in Progress Version 9.1C.

3.10.1 Handling Of Asynchronous COM Events

In Progress Version 9.1C, there is a new startup parameter, `-filterocxevents`, that handles asynchronous COM events differently than they were in previous versions. 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. To prevent asynchronous COM events from being handled during the second and third conditions, use the `-filterocxevents` startup parameter.

3.11 Progress Explorer Tool

The Progress Explorer Tool has been enhanced to incorporate the following updates being made for Progress Version 9.1D:

- A Register with NameServer check box that allows you the option of choosing a controlling NameServer to work with the Progress AppServer, the AppServer Internet Adapter (AIA), the Progress SonicMQ Adapter, the WebSpeed Transaction Server, WebSpeed Messengers, or the Progress DataServers. Prior to Version 9.1D, these Progress components were required to register with a NameServer.

  Depending on the way you configure and use the Progress product, you can make the decision about whether to specify a controlling NameServer to handle client connection requests. For example, your configuration might be simple and not require the location transparency the NameServer provides, or it might be more complex and, therefore, dependent on this and other NameServer features.

  If you choose not to register an AppServer Internet Adapter with a NameServer, then the AIA will be directly connected to an AppServer client and you will need to specify the host location and port of that AppServer client. Likewise, if you choose not to register a WebSpeed Messenger with the NameServer, then the Messenger will be directly connected to a WebSpeed Transaction Server, and you will need to specify the host location and port of that WebSpeed Transaction Server.

- Enhanced AdminServer security, such as:

  – User authentication and user authorization that the system administrator can optionally enable during the installation of Progress. This increased level of security is apparent in the Progress Explorer when you log on and again when you attempt the required connection to an AdminServer.

- Improved AdminServer log details, which allow you to see more descriptive information when a login to the Progress Explorer or a connection to the AdminServer fails. The log also contains details about AdminServer login and connection success; you can choose to limit the logging of this success information by using the new `DLogLevelSecurity` AdminServer command-line option if you are concerned about log size.

  For more information about this option, see:

Manual:	The "AdminServer Logging Enhancement" section in Appendix F, "Administration,"

- A check box in the SonicMQ Adapter General properties that allows you to start an Adapter instance automatically when the AdminServer starts. There is also a new Pool Range property in the SonicMQ Adapter Server properties that you use to specify settings for the number of threads within a SonicMQ Adapter broker process: Initial number of threads to start (the default is 20), Minimum threads (the default is 10), and Maximum threads (the default is 30).

- If you are running Fathom for Management Standard Edition, you also see some Fathom-specific fields in the Progress Explorer Tool. For example, when you set database configuration properties, you can choose whether to start the database monitoring agent automatically when the database starts.

- The WebSpeed Transaction Server, AppServer, and SonicMQ Adapter broker configurations now contain user name and password combinations. If your system administrator has established that you must enter your username and password to perform AdminServer operations, you must provide them in order to start, stop, or otherwise work with the AdminServer.

For more information, see the Progress Explorer Tool online help.

In Progress Version 9.1D, the Progress Explorer tool, is now a snap-in for the Microsoft Management Console (MMC). The Progress Explorer tool runs on the Windows 95, Windows 98, Windows 2000, and Windows NT platforms.

The following additional Progress Explorer tool enhancements are included in the 9.1C release:

- A new user interface within the MMC framework.
- Online, context-sensitive HTML help.

- Simultaneous connection to multiple AdminServers. This capability is important if you have Progress server products installed on more than one system. By supporting multiple AdminServers, Progress Explorer allows you to manage the Progress servers on each system through a single interface.
- The ability to view log files that record data specific to a server product or an instance. For example, a separate log file is created for each database; and each WebSpeed and AppServer broker instance has both a server log file and a broker log file. Additionally, the AdminServer has a log file named admserv.log that records status and error messages.

3.11.1 Starting the Progress Explorer Tool

To start the Progress Explorer from the Windows Desktop, choose Start→ Programs→ Progress→ Progress Explorer Tool. The Progress Explorer snap-in opens in the Microsoft Management Console (MMC).

For information about connecting to an AdminServer and working with the Progress Explorer tool, see the Progress Explorer tool online help. For more information about MMC, see the MMC online help.

3.12 Progress 4GL Enhancements

Please note the following changes in feature behavior:

- Copying MEMPTRs
- MEMPTRs as parameters

3.12.1 Copying MEMPTRs

In Version 9.0, when you copy one MEMPTR (M1) to another MEMPTR (M2), only the MEMPTR address is copied and both MEMPTRs point to the same memory location (L1).

In Version 9.1, when you copy one MEMPTR (M1) to another MEMPTR (M2), the memory location (L1), including the data, also gets copied. Therefore, MEMPTR M1 points to memory location L1, and MEMPTR M2 now points to memory location L2, which contains a copy of the data in L1.

This change has two ramifications for existing programs:

- In Version 9.0, you can change the data in the single memory location and both MEMPTRs will point to the changed data. In Version 9.1, you must change the data in both memory locations if you want both MEMPTRs to reflect the change.
- In Version 9.0, to clear memory after using the MEMPTRs, you could SET-SIZE = 0 on just one MEMPTR. In Version 9.1, you must SET–SIZE = 0 on both MEMPTRs to make sure that both memory locations are cleared.

3.12.2 MEMPTRs As Parameters

In Version 9.0, when you pass a MEMPTR to a remote procedure running on an AppServer, just the pointer value is passed.

In Version 9.1, when you pass a MEMPTR to a remote procedure running on an AppServer, the memory pointed to by the MEMPTR is transmitted to the remote procedure.

This change has two ramifications for existing programs:

- In Version 9.0, you can change the data in the single memory location and both MEMPTRs will correctly point to the changed data. In Version 9.1, you must change the data in both memory locations if you want both MEMPTRs to correctly reflect the change.
- In Version 9.0, to clear memory after using the MEMPTRs, you could SET–SIZE = 0 on just one MEMPTR. In Version 9.1, you must SET–SIZE = 0 on both the local and the remote MEMPTRs to make sure that both memory locations are cleared.

NOTE: The behavior of MEMPTRs passed as parameters within local procedures has not changed.

3.13 Progress SQL-92

The following sections describe Progress SQL-92 changes for Version 9.1:

- New Progress SQL-92 JDBC Driver

3.13.1 New Progress SQL-92 JDBC Driver

The JDBC driver in Version 9.0 has been replaced with a new industry standard JDBC driver from MERANT, Inc., formerly known as INTERSOLV. This new driver includes new JDBC class and library files. In order to use this new driver, you must change your environment variables and replace references to class and library files with the new class and library filenames:

- References to `dhjdbc.dll` should be changed to `JdbcProgress.dll`
- References to `libdhjdbc.so` should be changed to one of the following:
  - `libJdbcProgress.so` (Solaris, Compaq Tru64)
  - `libJdbcProgress.sl` (HP)
  - `libJdbcProgress.a` (AIX)
- References to ProgressDriver should be changed to `JdbcProgressDriver`, as in `com.progress.sql.jdbc.JdbcProgressDriver`.
- URL References to `jdbc:progress:T: . . .` should be changed to `jdbc:Jdbcprogress:T....`"

For more complete information, see:

Manual:	The *Progress JDBC Driver Guide*.

3.14 SmartObjects

Please note the following changes in feature behavior:

- Dynamic SmartDataObject Super-procedure Changes
- New Naming Convention for SmartObject Files
- Multiple Field Sources in Smart Display Objects
- Stateless SmartDataObjects
- SmartDataViewer Changes
- SmartDataBrowser Scroll-bar Behavior Changes

3.14.1 Dynamic SmartDataObject Super-procedure Changes

To support the Dynamic SmartDataObject, the behavior of three super-procedure routines has changed.

constructObject (containr.p)

Differences include:

- **Old behavior** — If there is a local database connection, then construct the complete object, else construct the client-only (`_cl`) object.
- **New behavior** — If there is a local database connection, then construct the complete object, else look for the client-only (`_cl`) object. If the client-only object is in the PROPATH search space, then construct it, else construct a dynamic object.

clientSendRows (was data.i, now data.p)

Differences include:

- **Old behavior** — Require a temp-table name as OUTPUT parameter.
- **New behavior** — Accept either a name or a handle of a temp-table as OUTPUT parameter. If a handle, create a dynamic version of the temp-table on the client.

submitRow (data.p)

Using the dynamic SmartDataObject shifts all validation to the AppServer. Turning off the AutoCommit property causes updates to be accumulated for manual commitment. These two factors will cause errors reported by *fieldname*Validate( ) and RowObjectValidate( ) not to be seen until the accumulated updates are finally committed.

3.14.2 New Naming Convention For SmartObject Files

Conventionally, certain SmartObjects have been given filenames beginning with a specific single lowercase letter, for example ‘d’ for SmartDataObjects. This worked well enough when there were fewer object types, and filenames were restricted by the old 11-character limit imposed by legacy operating systems. But now the filename restriction no longer exists, and it is increasingly difficult to find appropriate single characters for the new object types. For these reasons, Progress Software has begun to adopt a new naming convention.

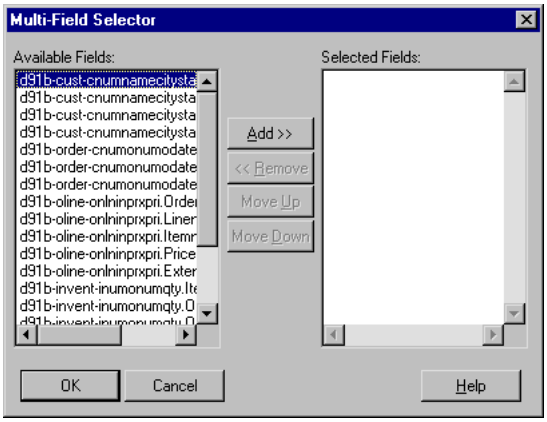
The new convention for naming object files uses three characters, beginning with the letter ‘s’. The objects to which it applies as of this release are:

- **sbo** — SmartBusinessObjects
- **smr** — Smart[Message]Receiver objects
- **sms** — Smart[Message][Sender objects
- **sbb** — SmartB2BObjects
- **sdf** — SmartDataField objects

Note that the existing prefixes remain unchanged at this time, with the exception of the prefix for SmartDataField object, which was ‘s’ by itself.

3.14.3 Multiple Field Sources In Smart Display Objects

In earlier releases, every field in a SmartDataViewer or SmartDataBrowser had to be supplied by the same SmartDataObject. With the new SmartBusinessObject, that restriction is no longer true. When you connect your Smart display object to a SmartBusinessObject, AppBuilder's SmartDataViewer Wizard will list the fields available from all the SmartDataObjects in that SmartBusinessObject:



You can then select any fields you like, effectively sourcing data on a per-field basis.

3.14.4 Stateless SmartDataObjects

In Version 9.0, all SmartDataObjects are run statefully, regardless of the operating mode of the AppServer Broker. Even though the Progress Explorer tool allows a site administrator to specify STATELESS mode for the AppServer Broker, SmartDataObjects ignore this specification and always run persistently, thus binding the AppServer session to one client for the life of the connection.

In Version 9.1, SmartDataObjects can sense the operating mode of the AppServer Broker and adjust accordingly. If STATELESS mode is specified for the AppServer Broker, the SmartDataObject, by default, runs in STATELESS mode as well, functioning exactly as it would if it were running in STATE-AWARE or STATE-RESET mode. The only noticeable difference is that each server request might take more time because the server-side SmartDataObject must be reestablished for each request. The benefit is that many clients can share fewer AppServer sessions with minimal performance degradation.

You might want to override this default behavior if, for example, the overhead of repeatedly reestablishing a complex query is so great that it degrades performance. To do this, check the Force to Stateful Operating Mode check box on the SmartDataObject's Instance Properties dialog box. By default, this box is unchecked, and the SmartDataObject inherits its operating mode from the AppServer. If you check this box, the SmartDataObject runs statefully, even if the AppServer Broker operating mode is set to STATELESS.

3.14.5 SmartDataViewer Changes

In Progress Version 9.0, the SmartDataViewer cannot be a container SmartObject.

In Version 9.1, the SmartDataViewer template `viewer.w` has been modified to activate a toggle box in the Advanced Procedure property sheet that allows you to drop other SmartObjects onto a SmartDataViewer. However, you cannot drop all types of objects onto a SmartDataViewer. A SmartDataViewer is intended to contain only basic objects, SmartDataObjects that will be used with SmartSelects, and/or SmartDataFields. (SmartSelects and SmartDataFields are new in Version 9.1.) You typically use the contained SmartDataFields to provide specially formatted displays of data in selected fields in the SmartDataViewer. For example, you might develop a SmartDataField that appears as a calendar and use it in a SmartDataViewer to display a date field.

3.14.6 SmartDataBrowser Scroll-bar Behavior Changes

To support navigation through the whole data stream using the Browser's scroll-bar thumb, a new ROW-DISPLAY event handler has been defined in browser.p. This changes the behavior of the object.

- **Old behavior** — If the user moved the scroll-bar thumb to the end of the scroll bar, the Browser would display the last record in the buffer and stop. This was confusing to many users, because the standard meaning of dragging the thumb to the end is end-of-document.

- **New behavior** — If the user moves the scroll-bar thumb to the end of the scroll bar, bringing the first/last record in the buffer into view, the Browser reads a new batch of records from the Data-Source if possible, and repositions the thumb to reflect the change. Thus, the thumb will never remain at the top or bottom of the scrollbar unless the first/last record in the whole data stream—not merely the first or last in the buffer—is in view.

3.15 WebClient Enhancements

Please note the following changes in feature behavior:

- Modifications to the WebClient Installer user interface for Version 9.1D
- Automatic Conversion of Configuration Files in 9.1C

3.15.1 Modifications To the WebClient Installer Process For Version 9.1D

In Progress Version 9.1D, the patch procedure and user interface associated with the Version 9.1C WebClient Installer program have been updated to provide end users a clearer, more direct means to install and confirm the installation of a WebClient patch during the WebClient installation process.

For specific details that compare the Version 9.1C WebClient Installer Process with the updated Version 9.1D WebClient Installer Process, see:

Manual:	Appendix G, “Progress WebClient” in this bulletin.

3.15.2 Automatic Conversion Of Configuration Files

The format of the application configuration (`.prowcapp`) file has changed from Version 9.1B. The Version 9.1C application deployer automatically converts any configuration file from an earlier version into the Version 9.1C format. After converting the configuration file into the new format, the application deployer follows the normal Version 9.1C procedure, using the WebClient Application Assembler, generating the application, etc.

A

Progress DataServer

This appendix documents enhancements and changes made to the DataServer and contains information about:

- Changes and Enhancements Made In Version 9.1D
- Changes and Enhancements Made In Version 9.1C
- Changes and Enhancements Made In Version 9.1B

A.1 Changes and Enhancements Made In Version 9.1D

Progress Version 9.1D provides new and enhanced functionality for the following Progress DataServer products:

- Progress Version 9.1D DataServer For ORACLE
- Progress Version 9.1D DataServer For Microsoft SQL Server
- Progress Version 9.1D DataServer For ODBC
- Progress/400 Product

A.1.1 Progress Version 9.1D DataServer For ORACLE

The Progress Version 9.1D DataServer for ORACLE contains the following enhancements over previous versions:

- Collation Table Support For ORACLE
- Include Default Option For ORACLE
- Alternative No NameServer Support For ORACLE
- Dynamic Find Feature For ORACLE
- Quoter 4GL Function For ORACLE

Collation Table Support For ORACLE

Version 9.1D of the Progress DataServer for Oracle now supports the user specifying the collation table that should be used through the Create/Modify Database Record for DataServer Schema. You can access the Collation option through the Data Administration tool, and select DataServer-> Oracle Utilities-> Create DataServer Schema. The dialog box shown in Figure A–1 opens.

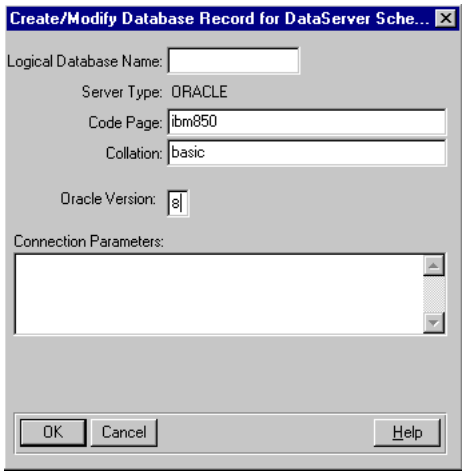


Figure A–1: Create/Modify Database Record For DataServer Schema Dialog Box

The collation table that the user specifies must be defined in the `convmap.dat` file or the user will receive an error message stating that the combination between the code page and the table does not exist.

The user can also specify the Collation table when performing the PROGRESS DB to Oracle Conversion. To access this utility, choose DataServer-> ORACLE Utilities-> Schema Migration Tools-> Progress DB to ORACLE. The dialog box shown in Figure A–2 opens.

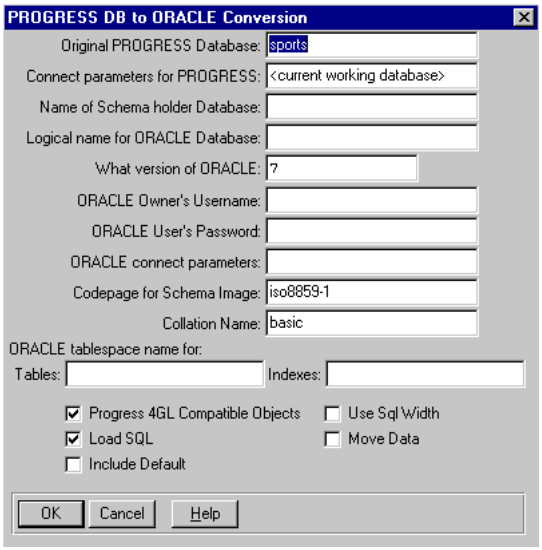


Figure A–2: PROGRESS DB To Oracle Conversion Dialog Box

For more information on collation tables, see the *Progress Internationalization Guide*.

Include Default Option For ORACLE

In Progress Version 9.1D, there is also an additional option in the ProtoOra Utility called the Include Default option. Selecting the Include Default option causes the conversion to include the default values for fields in the Oracle schema. The code uses the Progress initial value for varchar and numbers. Additionally, if the Progress date field has TODAY as the initial value, the default value in Oracle will be SYSDATE. The logical data type has either 1 or 0 depending on whether the initial value is set to TRUE or FALSE. The utility only supports True/False or Yes/No for logical initial values.

The new Include Default option is also available in the Delta df to Oracle Conversion utility. To access this utility, choose DataServer-> Oracle Utilities-> Schema Migration Tools-> Generate Delta.slq Progress to Oracle. The dialog box shown in Figure A–3 opens.

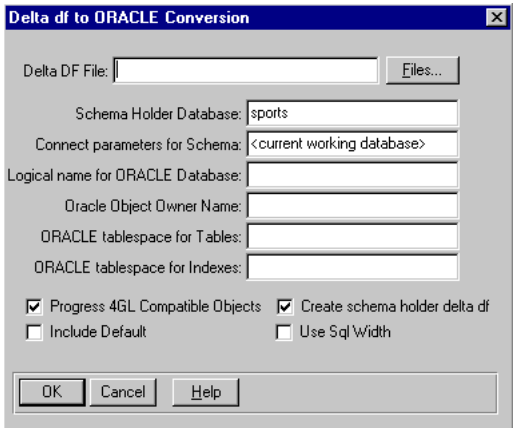


Figure A–3: Delta df To Oracle Conversion Dialog Box

Alternative No NameServer Support For ORACLE

In prior versions of the DataServer for Oracle, you can connect through the Progress Explorer without using the NameServer by providing a special value of "none" to the `-DataService` connection parameter. Now, in Progress Version 9.1D, to indicate that the NameServer should not be used, the DataServer for Oracle can use the `-DirectConnect` connection parameter. If the `-DirectConnect` connection parameter is specified, then the `-H` and `-S` parameters are interpreted as the host and port of the broker. However, unlike connecting directly to the broker with the `-DataService` connection parameter, using the `-DirectConnect` parameter makes the `-H` and `-S` parameters optional. If `-H` is not provided, it defaults to "localhost". If `-S` is not specified, it defaults to 4445. If `-DirectConnect` is specified, the `-DataService` is ignored.

Dynamic Find Feature For ORACLE

The Progress DataServer for Oracle in Version 9.1D will execute Dynamic Find statements like a FOR FIRST or FOR LAST statement. This should give much better performance than a typical FIND FIRST or FIND LAST statement. In particular, the NO-LOCK and SHARE-LOCK Dynamic Find statements will be executed using a single sql SELECT statement, instead of the two sql SELECT statements used for a typical FIND statement.

Quoter 4GL Function For ORACLE

The quoter 4GL function can be useful for quoting values or handling embedded quotes when building SQL statements that will be passed into the send-sql-statement stored procedure. For more information on the quoter function, see Appendix D, "Progress 4GL Language Reference" in this bulletin and the *Progress Programming Handbook*.

A.1.2 Progress Version 9.1D DataServer For Microsoft SQL Server

The Progress Version 9.1D DataServer for Microsoft SQL Server has the following enhancements:

- Collation Table Support For Microsoft SQL Server
- Alternative No NameServer Support For Microsoft SQL Server r
- Insert Trigger Support For Microsoft SQL Server
- Dynamic Find Feature For Microsoft SQL Server
- Quoter 4GL Function For Microsoft SQL Server
- FOR FIRST and FOR LAST Performance For Microsoft SQL Server
- DataServer (–Dsrv) Startup Parameter Option For Microsoft SQL Server
- Block Cursor For Microsoft SQL Server

Collation Table Support For Microsoft SQL Server

Version 9.1D of the DataServer for Microsoft SQL Server allows the user to specify the collation table that should be used through the Create/Modify Database Record for MSS Schema dialog box. You can access the Collation option through the Data Administration tool, and select DataServer-> MSS Utilities-> Create DataServer Schema. The dialog box shown in Figure A–4 opens.

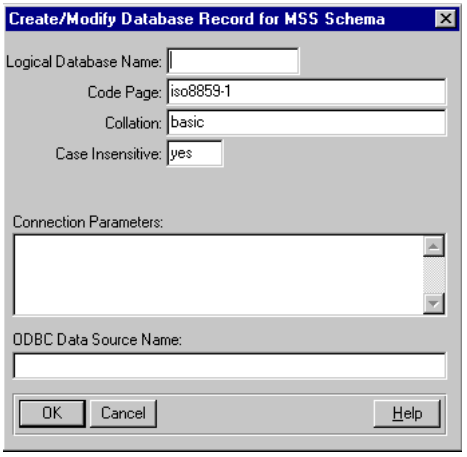


Figure A–4: Create/Modify Database Record For MSS Schema Dialog Box

The collation table that the user specifies must be defined in the `convmap.dat` file or the user will receive an error message stating that the combination between the code page and the table does not exist.

There is also a new fill-in, Case Insensitive. The case-insensitive selection is to determine if the code page used in the MS SQL Server database engine, the code page selected during installation of the SQL Server product, is case insensitive. MS SQL Server is case insensitive by default, so there is no need to add shadow columns, but if the code page is sensitive, then the user has the option to create shadow columns.

For more information on collation tables, see the *Progress Internationalization Guide*.

The user can also specify the Collation table in the PROGRESS DB to MS SQL Server Conversion. To access the utility from the Data Administration tool, choose DataServer-> MS SQL Server Utilities-> Schema Migration Tools-> PROGRESS DB to MS SQL Server. The dialog box shown in Figure A–5 opens.

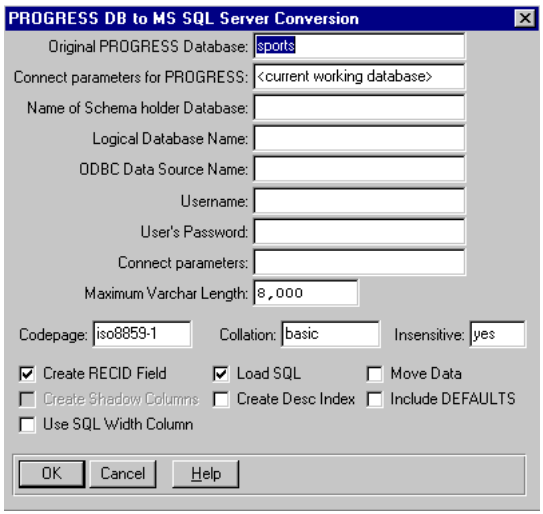


Figure A–5: PROGRESS DB To MS SQL Server Conversion Dialog Box

Alternative No NameServer Support For Microsoft SQL Server

In prior versions of the DataServer for Microsoft SQL Server, you can connect through the Progress Explorer without using the NameServer by providing a special value of "none" to the `-DataService` connection parameter. Now, in Progress Version 9.1D, to indicate that the NameServer should not be used, the DataServer for Microsoft SQL Server can use the `-DirectConnect` connection parameter. If the `-DirectConnect` connection parameter is specified, then the `-H` and `-S` parameters are interpreted as the host and port of the broker. However, unlike connecting directly to the broker with the `-DataService` connection parameter, using the `-DirectConnect` parameter makes the `-H` and `-S` parameters optional. If `-H` is not provided, it defaults to "localhost." If `-S` is not specified, it defaults to 4446. If `-DirectConnect` is specified, the -DataService is ignored.

Insert Trigger Support For Microsoft SQL Server

Progress Version 9.1D has modified the insert trigger that it places on a table in the foreign database when the Progress RECID support is selected. The modification reduces the level of contention between multiple users performing inserts simultaneously. The reduction in contention can produce performance improvement.

If you perform a migration using the MS SQL Server Progress DB to MS SQL Server utility and select to have Progress RECID support, Progress recommends that you dump your data, re-create your MS SQL Server database by redoing the migration and reload the data. If this is not possible, then you can manually change your existing MS SQL Server database by doing the following on each table that has been defined with Progress RECID support from your SQL Server Migration:

1 ♦ Drop the unique index created over the progress_recid field. There should be one index for each table created during the initial migration. The syntax is:

SYNTAX

```
DROP index <database>.<owner>.<table name>#_#progress_recid.
```

2 ♦ Drop the old insert trigger. The name typically looks like _TI_<tablename>. The syntax is:

SYNTAX

```
DROP trigger <database>.<owner._TI_<table name>
```

3 ♦ Add the new trigger. The syntax is:

```
CREATE TRIGGER _TI_<table name> ON <table name> FOR INSERT AS
    IF ( SELECT PROGRESS_RECID FROM INSERTED ) IS NULL
    BEGIN
       UPDATE t SET PROGRESS_RECID = i.IDENTITYCOL
          FROM <table name> t JOIN INSERTED i ON
          t.PROGRESS_RECID_IDENT_ = i.PROGRESS_RECID_IDENT_
          SELECT CONVERT (int,@@identity)
    END
```

Sample code for db.owner.Customer table is:

```
CREATE TRIGGER db.owner._TI_customer ON customer FOR INSERT AS
   IF ( SELECT PROGRESS_RECID FROM INSERTED ) IS NULL
   BEGIN
      UPDATE t SET PROGRESS_RECID = i.IDENTITYCOL
      FROM customer t JOIN INSERTED i ON
      t.PROGRESS_RECID_IDENT_ = i.PROGRESS_RECID_IDENT_
      SELECT CONVERT (int, @@identity)
   END
```

4 ♦ Create a new **nonunique** index for each table naming the new index the same name as was dropped earlier in Step 1. Make sure it is not unique. The syntax is:

```
CREATE INDEX <database>.<owner>.<table>#_#progress_recid ON
<table name>(PROGRESS_RECID)
```

A second index must be created for the `progress_recid_ident` field. This index must be created as **unique** and named `<table name>#_#progress_recid_ident`. The syntax is:

```
CREATE UNIQUE INDEX <table name>#_#progress_recid_ident_ ON
<table name>(PROGRESS_RECID_IDENT_
```

An example of the two indices that must be created for the customer table is as follows:

```
CREATE INDEX customer#_#progress_recid ON CUSTOMER (PROGRESS_RECID)

CREATE UNIQUE INDEX customer#_#progress_recid_ident_ ON
CUSTOMER(PROGRESS_RECID_IDENT_)
```

Dynamic Find Feature For Microsoft SQL Server

In Progress Version 9.1D, the DataServer for Microsoft SQL Server executes Dynamic Find statements like a FOR FIRST or FOR LAST statement. This should provide improved performance over a typical FIND FIRST or FIND LAST statement. In particular, the NO-LOCK Dynamic Find statements will be run using a single SQL SELECT statement instead of the two SQL SELECT statements required by a typical FIND statement. In addition, the result set is limited to a single record, as they are with a FOR FIRST and a FOR LAST.

Quoter 4GL Function For Microsoft SQL Server

The quoter 4GL function can be useful for quoting values or handling embedded quotes when building SQL statements that will be passed into the send-sql-statement stored procedure. For more information on the quoter function, see Appendix D, "Progress 4GL Language Reference" in this bulletin and the *Progress Programming Handbook*.

FOR FIRST and FOR LAST Performance For Microsoft SQL Server

The FOR FIRST and FOR LAST functions in the Progress Version 9.1D DataServer for Microsoft SQL Server has been enhanced by limiting the server-side result set to a single record. This should provide additional improvement in performance for these statements over traditional FIND statements. The new dynamic find statements emulate the new FOR FIRST and FOR LAST behavior and also provide performance improvement over traditional FIND statements.

DataServer (–Dsrv) Startup Parameter Option For Microsoft SQL Server

RECID Buffer: The ZPRGRS_RECID_BUF_SIZE,xxx Option

If you do not select the Progress RECID field in your Progress-to SQL Server Database conversion, then a unique index over each table is selected to support Progress RECID/ROWID functions. RECID functionality enables backward and forward scrolling in the DataServer for Microsoft SQL Server product.

In Progress Version 9.1D, the RECID buffer is used to store key information about the unique index selected to support RECID. The default RECID buffer size is 245 bytes. The space is used dynamically to allocate the RECID buffer for a given record. The RECID buffer size needs to be large enough to contain all the key information regarding the unique index. If the unique index selected is a composite of many columns and/or contains large column names, the RECID key information might exceed the RECID buffer size and issue an error message 2090.

If you exceed the RECID buffer limit, Progress first recommends that you try to find an alternative unique index with a smaller number of key fields to represent the Progress RECID. This should help improve performance during record access as well. You can change the RECID index in the Progress dictionary by selecting the "DataServer" button from the Table Properties of the table in your schema holder.

If an alternative unique index is not available and it is not possible to reduce the key size of your existing RECID unique index to avoid overrunning the buffer, you can expand the area by setting the startup option as follows:

"-Dsrv ZPRGRS_RECID_BUF_SIZE,xxx"

Where *xxx* is the new size of the area in bytes.

You can expand the RECID buffer to a maximum of 1000 bytes. The minimum size is 44 bytes, which is the size required by the PROGRESS_RECID field when the Progress RECID field is selected during database conversion. The algorithm to determine your adjusted size is as follows:

9 + Length of first Index column name + Length of first column data + 17

+ Length of second Index column name + Length of second column data + 22

+ Length of third Index column name + Length of third column data + 22

+ ...

Block Cursor For Microsoft SQL Server

In Progress Version 9.1D, block cursors are available as a performance enhancement. They allow blocks of records to be bound and fetched together from a single database request, thus reducing the network overhead and minimizing record copies for 4GL queries and finds, and minimizing record copies. In Progress Version 9.1D, block cursors become the default behavior replacing look-ahead cursors for NO-LOCK queries. A NO-LOCK query is one that executes with the NO-LOCK lock condition attached to its 4GLstatement. DataServer connections that run at the read uncommitted isolation level will also execute queries that specify the SHARE-LOCK condition as NO-LOCK queries and thus will also utilize block cursors for their result sets. You can shut off block cursor behavior and revert back to look-ahead cursors for NO-LOCK queries by setting the new -Dsrv parameter PRGRS_BLOCK_CURS to zero. If you leave block cursors on but wish to turn off the block and/or look-ahead cursor optimizations for a specific query, you can set the QUERY-TUNING option QUERY-TUNING(NO-LOOKAHEAD) on your 4GL statement.

While block cursors increase efficiency, they might also require more memory than look-ahead cursors. Block cursors will try to reuse memory from previous result sets whenever possible. However, you can adjust the memory available to block cursors according to your specifications. Adjust the cache size of an individual block with the same connection-level parameter you used to set the look-ahead cache size. The QT_CACHE_SIZE -Dsrv option allows you to specify at connect time how large the block cache should be for individual query results. When block cursors are active, the default QT_CACHE_SIZE is set to 10,000 bytes (when look-ahead cursors are active, this size defaults to 30,000 bytes instead).

You can override the connection-level QT_CACHE_SIZE at the statement level by setting the QUERY-TUNING option called CACHE-SIZE on the query itself. Determining the proper block size should be based on the maximum length of your returned records multiplied by the expected number of resultant rows and then compared to your available memory. Whenever an existing block is not reused, the block of memory allocated for the query will be adjusted downward to an exact multiple of the number of rows that can fit into the allocated area.

To prevent large applications from potentially overutilizing memory for block cursors, two other settings can be adjusted to limit the amount of memory available to block cursors. The first is the maximum block memory usage per table. If you have multiple queries open against the same table simultaneously, each query uses its own query block. The PRGRS_TABLE_BLOCKSIZE -Dsrv option puts an upper limit on the total memory available to query blocks of an individual table. This number should normally be set larger than, and perhaps as a multiple of, the QT_CACHE_SIZE value. If the maximum block memory available to the table is, or would be, exceeded by allocating space for the current NO-LOCK query in your 4GL, the query is instead executed with a look-ahead cursor. The default maximum block memory area per table is set at 65,000 bytes.

The second adjustment switch available is the PRGRS_MAX_BLOCKSIZE -Dsrv option. This value sets an upper limit to the overall memory provided to block cursors (irrespective of per table allocations). The default maximum is 1048576 bytes (or 1MB). Your accumulated total memory allocated for block cursors will not exceed this value. If allocating block cursor memory for a NO-LOCK query in your 4GL would cause this limit to be exceeded, the query would instead be executed with a look-ahead cursor. The PRGRS_MAX_BLOCKSIZE value should be set with respect to the amount of memory available on your client machine when running the Personal edition of the Microsoft SQL Server DataServer. If you are running in Enterprise mode using the Enterprise edition, the number should be set with respect to both the available memory on the server machine as well as the number of Progress clients that will be connecting through your Progress database.

The following example uses the state table in the demo database, which has a maximum record size of 84 bytes:

```
FOR EACH state NO-LOCK QUERY-TUNING(CACHE-SIZE 850):
      DISPLAY state.
END.
```

In this example, the QUERY-TUNING CACHE-SIZE value overrides the -Dsrv QT_CACHE_SIZE query block size default of 10,000 bytes. Ten records at 84 bytes per record would total 840 bytes in the query block. If a previously established block was located for that table that was greater than or equal to the requested size, it would be reused. However, if a new block were established instead, its memory allocation would be adjusted downward to 840 bytes in order to fit an exact multiple of rows in the block. If the result set contained 30 records, for example, the query block would get refilled three times before the end of the result set if read sequentially

A.1.3 Progress Version 9.1D DataServer For ODBC

The Progress Version 9.1D DataServer for ODBC contains the following enhancement over previous versions:

- Collation Table Support For ODBC
- Alternative No NameServer Support For ODBC
- Dynamic Find Feature For ODBC
- Quoter Function For ODBC
- DataServer (–Dsrv) Startup Parameter Option For ODBC

Collation Table Support For ODBC

Version 9.1D of the DataServer for ODBC now allows the user to specify the collation table that should be used through the Create/Modify Database Record for DataServer Schema dialog box. You can access the Collation option through the Data Administration tool, and select DataServer-> ODBC Utilities-> Create DataServer Schema. The dialog box shown in Figure A–6 opens.

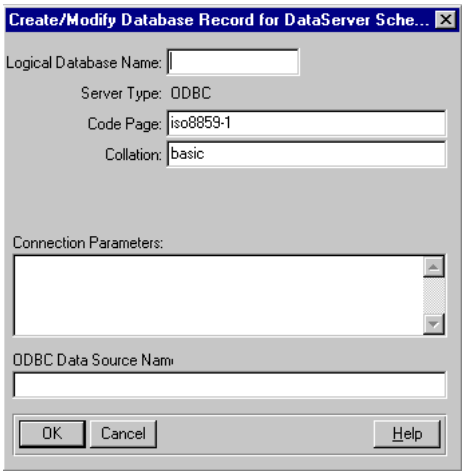


Figure A–6: Create/Modify Database Record For DataServer Schema Dialog Box

The collation table that the user specifies must be defined in the `convmap.dat` file or the user will receive an error message stating that the combination between the code page and the table does not exist.

For more information on collation tables, see the *Progress Internationalization Guide*.

The user can also specify the Collation table in the PROGRESS DB to ODBC Conversion screen. To access this utility from the Data Administration tool, choose DataServer-> ODBC Utilities-> Schema Migration Tools-> PROGRESS DB to ODBC. The dialog box shown in Figure A–7 opens.

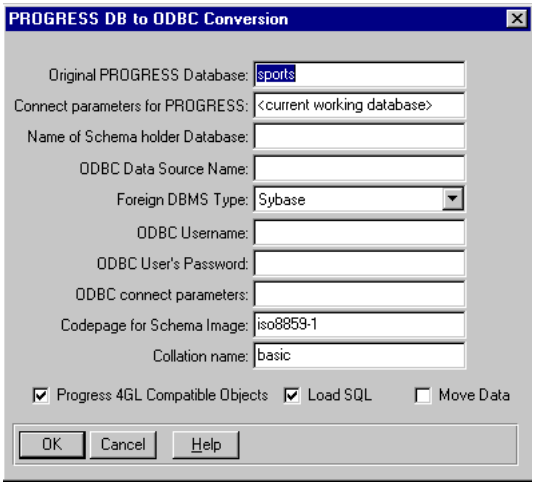


Figure A–7: PROGRESS DB To ODBC Conversion Dialog Box

Alternative No NameServer Support For ODBC

In prior versions of the DataServer for Microsoft ODBC, you can connect through the Progress Explorer without using the NameServer by providing a special value of "none" to the `-DataService` connection parameter. Now, in Progress Version 9.1D, to indicate that the NameServer should not be used, the DataServer for ODBC can use the `-DirectConnect` connection parameter. If the `-DirectConnect` connection parameter is specified, then the `-H` and `-S` parameters are interpreted as the host and port of the broker. However, unlike connecting directly to the broker with the `-DataService` connection parameter, using the -DirectConnect parameter makes the `-H` and `-S` parameters optional. If `-H` is not provided, it defaults to "localhost". If `-S` is not specified, it defaults to 4444. If `-DirectConnect` is specified, the `-DataService` is ignored.

Dynamic Find Feature For ODBC

In Progress Version 9.1D, the DataServer for ODBC executes Dynamic Find statements like a FOR FIRST or FOR LAST statement. This should provide increased performance over a typical FIND FIRST or FIND LAST statement. In particular, the NO-LOCK Dynamic Find statements are executed using a single SQL SELECT statement instead of the two SQL SELECT statements required by a typical FIND statement.

Quoter Function For ODBC

The quoter function can be useful for quoting values or handling embedded quotes when building SQL statements that will be passed into the send-sql-statement stored procedure. For more information on the quoter function, see Appendix D, "Progress 4GL Language Reference" in this bulletin and the *Progress Programming Handbook*.

DataServer (–Dsrv) Startup Parameter Option For ODBC

RECID Buffer size: The ZPRGRS_RECID_BUF_SIZE,xxx Option

The Progress Version 9.1D DataServer for ODBC tries to select a unique index over each table defined in your schema holder to support the Progress RECID/ROWID functions. RECID functionality enables backward and forward scrolling in the DataServer product.

The RECID buffer is used to store key information about the unique index selected to support RECID. The default RECID buffer size is 245 bytes. The space is used dynamically to allocate the RECID buffer for a given record. The RECID buffer size needs to be large enough to contain all the key information regarding the unique index. If the unique index selected is a composite of many columns and/or contains large column names, the RECID key information might exceed the RECID buffer size and issue error message 2090.

If you exceed the RECID buffer limit, Progress first recommends that you try to find an alternative unique index with a smaller number of key fields. This should help to improve performance during record access as well. You can change the RECID index in the Progress dictionary by selecting the "DataServer" button from the Table Properties of the table in your schema holder.

If it is not possible to change the selected RECID unique index for the table that is over running the buffer, you can expand the area by setting the startup option as follows:

```
"-Dsrv ZPRGRS_RECID_BUF_SIZE,xxx"
```

where xxx is the new size of the area in bytes.

You can expand the RECID buffer size to a maximum of 1000 bytes. The minimum size is 44 bytes. The algorithm to determine your adjusted size is as follows:

9 + Length of first Index column name + Length of first column data + 17

+ Length of second Index column name + Length of second column data + 22

+ Length of third Index column name + Length of third column data + 22

+ ...

A.1.4 Progress/400 Product

You can connect the following clients to the Progress/400 DataServer:

- 4GL Client (Character or GUI)
- WebSpeed
- AppServer

The follow chart shows the AS/400 client/server compatibility. Any combination not listed on this chart is not supported.

With the AS/400 DataServer, the following clients are supported:

Client Version	AS/400 DataServer Version
9.1D	9.1C
3.1D WebSpeed	9.1C

With the AS/400 AppServer, the following client is supported:

Client Version	AS/400 AppServer Version
9.1D	9.1C

To administer the Version 9.1C AS/400 AppServer/NameServer, you must use the Version 9.1C Progress Explorer.

A.2 Changes and Enhancements Made In Version 9.1C

- Progress Version 9.1C DataServer For Microsoft SQL Server
- Progress Version 9.1C DataServer For ODBC

A.2.1 Progress Version 9.1C DataServer For Microsoft SQL Server

The Progress Version 9.1C DataServer for Microsoft SQL Server had enhancements and new features of which the following -Dsrv startup parameters are documented here.

Query Result Order: The PRGRS_ALWAYS_INDEX Option

The default value for the PRGRS_ALWAYS_INDEX option is 1 or ON. For the Microsoft SQL Server DataServer, if MSS_PRESERVER_CURS is set to 1, then PRGRS_ALWAYS_INDEX can be set to zero. If MSS_PRESERVE_CURS is set to 0, then the PRGRS_ALWAYS_INDEX setting is ignored.

Some queries can gain a performance advantage when the result set from the query does not use an index to order the results. In older versions of the DataServer for Microsoft SQL Server, results sets were always indexed because queries and their results were gathered in segments with new cursors being generated for each query segment. However, most advanced drivers and data sources now allow a single cursor to be retained for the life of a given query, even across a transaction boundary. Drivers that can preserve cursors are allowed to execute unindexed queries if the PRGRS_ALWAYS_INDEX option is set to 0 (off). When this option is off and the user does not specify an index or BY clause for the results of their 4GL statement and the particular 4GL being executed does not require scrolling capability in the Progress client, the query result will remain unindexed by the DataServer.

NOTE: In an unindexed query, the DataServer will not send an ORDER BY clause to the data source. Regardless of whether or not an ORDER BY clause is used, the DataServer has no control over the index utilized by the data source.

Preserve Cursors: The MSS_PRESERVE_CURS,1 Option

The default value for the MSS_PRESERVE_CURS option is 0 or OFF.

The MMS_PRESERVERE_CURS,1 -Dsrv startup option allows server side cursors in Microsoft SQL Server to be preserved beyond a transaction boundary. Under many circumstances, preserved cursors can result in a significant performance improvement. However, in some instances, degradation has also been experienced as the result of this setting, so the default value for the -Dsrv switch toggle is off. If you want to turn it on to see if performance benefits are available to you through this setting, enter the following -Dsrv option:

-Dsrv MSS_PROSERVE_CURS,1.

A.2.2 Progress Version 9.1C DataServer For ODBC

The Progress Version 9.1C DataServer for ODBC had enhancements and new features of which the following -Dsrv startup parameter is documented here.

Query Result Order: The PRGRS_ALWAYS_INDEX Option

The default value for the PRGRS_ALWAYS_INDEX option is 1 or ON. For the ODBC DataServer, if the driver is capable of and is configured to allow its ODBC cursors to be preserved across transaction boundaries, then PRGRS_ALWAYS_INDEX can be set to zero. If cursor preservation is not allowed for the specified driver and/or not allowed with the data source to which the driver is connected, then the PRGRS_ALWAYS_INDEX setting is ignored.

Some queries can gain a performance advantage when the result set from the query does not use an index to order the results. Most advanced drivers and data sources now allow a single cursor to be retained for the life of a given query, even across a transaction boundary. Drivers that can preserve cursors are allowed to execute unindexed queries if the PRGRS_ALWAYS_INDEX option is set to 0 (off). See documentation for your ODBC data source and driver to determine if cursor preservation is possible. When this option is off and the user does not specify an index or BY clause for the results of their 4GL statement and the particular 4GL being executed does not require scrolling capability in the Progress client, the query result will remain unindexed by the DataServer.

NOTE: In an unindexed query, the DataServer will not send an ORDER BY clause to the data source. Regardless of whether or not an ORDER BY clause is used, the DataServer has no control over the index utilized by the data source.

A.2.3 Changes and Enhancements Made In Version 9.1B

Version 9.1B provides the following new changes and enhancements to the following Progress DataServer products:

- Progress DataServer For ORACLE
- Progress/400 Product
- Progress DataServer For Microsoft SQL Server

A.2.4 Progress DataServer For ORACLE

The DataServer for ORACLE contains the following enhancements over previous versions:

- RAW–TRANSFER Statement
- Bulk Insert In Schema Migration Tools
- GET PREVIOUS After Index Reposition

RAW–TRANSFER Statement

The DataServer for ORACLE supports the RAW–TRANSFER statement, which allows you to manipulate entire rows without any conversion. You can thus store an entire ORACLE database row in a Progress RAW field or in an ORACLE RAW (or RAW Long) column, then restore the field or column as an ORACLE database row. In a similar way, you can store a Progress database record in an ORACLE RAW (or RAW Long) column, then restore the column as a Progress database record. For more information on the RAW–TRANSFER statement, see the *Progress Programming Handbook*.

Bulk Insert In Schema Migration Tools

In the Schema Migration Tools for ORACLE, Progress now provides a Bulk Inserts utility. This utility allows the DataServer to insert many rows at a time into ORACLE, thus increasing performance.

GET PREVIOUS After Index Reposition

GET PREVIOUS after index reposition will now retrieve rows prior to the point to which you repositioned your query.

A.2.5 Progress/400 DataServer

The Progress/400 DataServer now includes enhancements to the `Load Data Definitions` tool.

Load Data Definitions Tool

There is a new schema migration option for the Progress/400 DataServer, "Load Data Definitions." You can access this option from the Data Administration Tool or the Character Data Dictionary. Choose DataServer→ DB2/400→ Schema Migration Tools→ Load Data Definitions to start the tool.

The tool enables any client to push schema through a definitions file to the Progress/400 dictionary library. In previous versions, only Windows clients could perform this function through the Progress/400 Data Administration utility.

The "Load Data Definitions" tool can load a definitions file in either Progress or AS/400 format. The utility places the user into Database Administrator (DBA) mode, loads the definition file, and commits the information.

A.2.6 Progress DataServer For Microsoft SQL Server

The DataServer for Microsoft SQL Server provides enhancement to the new DataServer (`-Dsrv`) startup parameter options.

DataServer (–Dsrv) Startup Parameter Options

Query Result Order: The PRGRS_ALWAYS_INDEX Option

Some queries can gain a performance advantage when the result set from the query does not use an index to order the results. In older versions of the DataServer for Microsoft SQL Server, results sets were always indexed because queries and their results were gathered in segments with new cursors being generated for each query segment. However, most advanced drivers and data sources now allow a single cursor to be retained for the life of a given query, even across a transaction boundary. Drivers that can preserve cursors are allowed to execute unindexed queries if the PRGRS_ALWAYS_INDEX option is set to 0 (off). When this option is off and the user does not specify an index or BY clause for the results of their 4GL statement and the particular 4GL being executed does not require scrolling capability in the Progress client, the query result will remain unindexed by the DataServer.

NOTE: In an unindexed query, the DataServer will not send an ORDER BY clause to the data source. However, the DataServer has no control over whether or not the actual data source utilizes an index in order to generate a result set.

Concurrent Query Execution: The PRGRS_CACHE_CONN Option

It is possible to run read-only queries in separate connections through the driver to the data source. Opening a separate connection to run a query or stored procedure might provide better performance; although, having to many open connections might also degrade performance. The PRGRS_CACHE_CONN option allows you to set a limit for the maximum number of server connections available in the Progress DataServer session. If the session attempts to exceed the maximum threshold, a session will need to wait for an existing connection to complete first before an additional connection can be made.

Concurrent Procedure Results: The PRGRS_PROC_TRAN Option

The DataServer allows only one active request for running a sorted procedure before you run a second one. However, you can process results from several stored procedures concurrently if you set the PRGRS_PROC_TRAN switch to 1 (on). When switched on, this option will cause a separate connection to be used for each stored procedure request (assuming the PRGRS_CACHE_CONN threshold has not been exceeded).

CAUTION: When procedures run in separate connections of the same DataServer session, the scopes of their respective transactions are isolated from one other. If one running procedure attempts to update the same record used by another running procedure in the same DataServer session, a lock time out or even a deadlock could occur.

Cursor Characteristics: The PRGRS_STABLE_CURS Option

The SQL Server driver and its data source ultimately determine whether a cursor is stable during the commit or rollback of a transaction and if the cursor can persist beyond a single transaction scope. The DataServer normally resolves cursor characteristics by interrogating the driver and setting the run-time environment accordingly. Progress does not recommend bypassing normal operation under most circumstances. However, if your driver and/or data source do not provide stable cursors, then under very specific circumstances, you might improve performance by setting the PRGRS_STABLE_CURS option to 1 (on). If all the 4GL you run in your Progress DataServer session is without transactions or if all of its queries and resultant data sets are fully processed on one or the other side of any existing transaction boundaries, setting this option is possible.

NOTE: This is a session-level switch which means all 4GL run in that session must comply with the above requirements.

Wait Time for Asynchronous Connections: The PRGRS_WAIT_DELAY Option

The PRGRS_WAIT_DELAY option is ignored unless you are running in SQL Server asynchronous mode. Check your driver and Progress connection requirements to determine if you are allowed to run in asynchronous mode. This option allows you to determine the number of seconds you would want the DataServer to delay further execution while waiting for the driver to respond to a request that might initiate a lock on database resources. The delay is initiated when the NO–WAIT option is used in your 4GL and the DataServer is awaiting a response to a database request made through the SQL Server driver

B

R-code Features and Functions

R-code is the intermediate binary code that Progress generates when it compiles 4GL source files. This is the code that Progress actually runs when it executes a procedure, whether from session compiles or from permanently generated r-code (.r) files.

Progress provides a dynamic r-code execution environment, as well as integrity and security mechanisms to ensure that your r-code is running in a compatible environment.

This appendix provides information about the following topics:

- R-code Structure
- R-code Libraries
- R-code Libraries and PROPATH
- R-code Execution
- R-code Portability
- Code Page Compatibility
- Database CRCs and Time Stamps
- R-code CRCs and Procedure Integrity

B.1 R-code Structure

R-code is divided into multiple segments of varying length. Each r-code file contains an object header and segment location table followed by the actual r-code segments. The object header is a fixed-length descriptor that identifies the file as an r-code file and contains information about the version and size of the r-code file. The segment location table is a variable-length descriptor that contains the size and location of each r-code segment in the file. The maximum size for most segment types is 62K.

Table B–1 describes and lists the types, maximum size, and maximum number of segments in an r-code file.

Table B–1: R-code Segments *(1 of 2)*

Segment Type	Max. Size	Max. Number	Description
Action code	62K	4 for the main procedure, 1 per internal procedure	Holds the actual executable code in the form of action cells. Action cells drive the Progress interpreter and contain the executable code for 4GL verbs. There are between one and four action code segments for the main procedure, and one separate action code segment for each internal procedure. **NOTE:** Code generated with multiple main action code segments can be run only with Progress Version 9.1C or later.
Expression code	62K	4	Holds the executable expressions in Reverse Polish Notation (RPN) for the main procedure, all internal procedures, and all trigger blocks. An r-code file can have up to 248K of executable expressions.
Text	62K	1 per language	Holds all literal character strings for the r-code file. There is one text segment for the default language and one for each language specified in the COMPILE statement. Duplicate literals are removed. Only one text segment is loaded into memory at run time.

Table B–1: R-code Segments *(2 of 2)*

Segment Type	Max. Size	Max. Number	Description
Initial value	62K	1	Contains information required to initialize an r-code file for execution, including database and table references, variable definitions, TEMP–TABLE and WORK–TABLE definitions, a list of defined frames, time stamps, and CRCs. There is one initial value segment per r-code file.
Frame	32K	1 per frame	Contains layout information for a frame, including frame fields, attributes, and all RPN expression code from the frame phrase. There is one frame segment for each named and unnamed frame in the r-code file. Each frame segment has a 32K limit.
Debugger	62K	1	Used by the Application Debugger to maintain the debugger context for the procedure. There is one debugger segment per r-code file. This segment is loaded into memory on demand only when the Debugger is active. For information about the Debugger, see the *Progress Debugger Guide*.

B.1.1 Factors That Affect R-code Size

You can affect r-code size by how you write your 4GL, depending on the r-code segment. The action code and initial value segments are among the most tunable.

Action Code Segment

You can reduce the size of an action code segment (for either the main procedure or an internal procedure) by consolidating multiple 4GL statements into one. This can also increase the speed of execution, because the interpreter executes only one action instead of several.

For example, you can reduce action code size by combining several consecutive assignment statements into one ASSIGN statement. The following examples require the same amount of expression code:

```
a = 1.
b = 2.
c = 3.
```

```
ASSIGN a = 1
       b = 2
       c = 3.
```

NOTE: The compiler creates up to four main action code segments, as needed. When the compiler creates multiple main action code segments, it generates a warning message. If, during compilation, the main action code segments exceed the total size of all four segments, the compilation fails.

Initial Value Segment

You can reduce the size of the initial value segment by limiting the number of SHARED variables that are accessed in a procedure. The r-code required to support a SHARED variable is larger than the r-code to support a NEW SHARED variable. Progress uses approximately 36 additional bytes in the initial value segment to resolve each SHARED variable at run time. This value can change depending on the environment.

B.1.2 R-code File Segment Layout

Figure B–1 shows the segment layout in an r-code file. A compiled procedure requires one initial value segment, at least one main action code segment, one expression code segment, one text segment, and one debugger segment. There can be more action and expression code segments, up to the limit, as required by your procedure. There can be multiple frame segments, one segment for each frame in your procedure. The maximum number of frame segments is virtually unlimited.

There can also be multiple text segments, one segment for the default language and each language that you choose for translation. The default language segment contains the literal character strings defined in the original 4GL source file. You can create additional text segments by using the Translation Manager. Although multiple text segments are possible, only one text segment is available per language. Thus, the literal character strings for a given language cannot exceed 62K. For more information about natural language translation for 4GL procedures, see the *Progress Translation Manager Guide*.

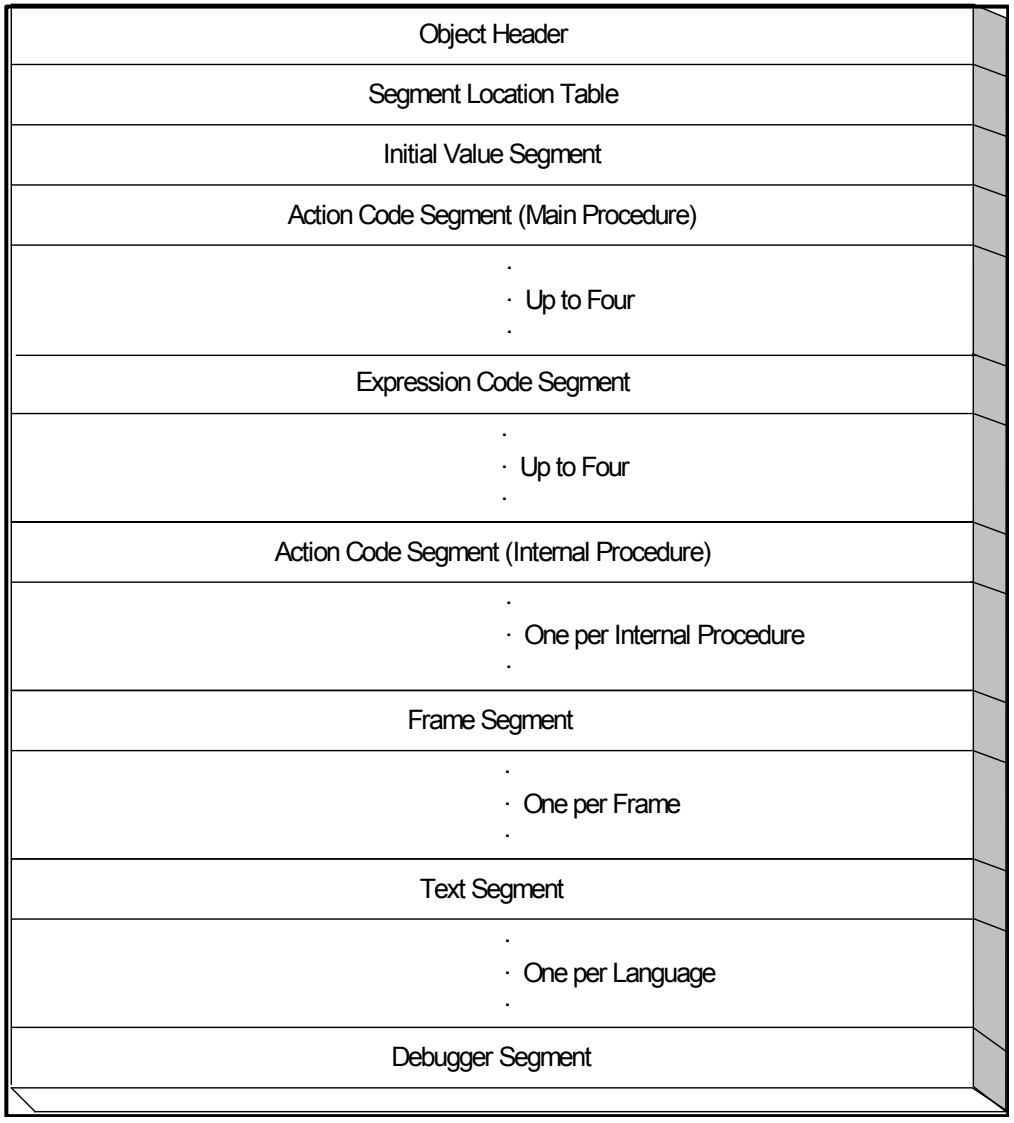


Figure B–1: R-code File Segment Layout

B.2 R-code Libraries

You can organize and store r-code files in a Progress r-code library. A Progress r-code library is a collection of r-code procedures combined in a single file. R-code libraries allow you to manage and execute r-code procedures more efficiently. You create r-code libraries by using the PROLIB utility.

Progress provides two types of r-code libraries: standard and memory-mapped. A standard library contains r-code procedures that execute in local memory. A memory-mapped library contains r-code procedures that execute in shared memory.

For information about using the PROLIB utility to create r-code libraries, see the *Progress Client Deployment Guide*.

B.3 R-code Libraries and PROPATH

The following rules govern how standard and memory-mapped libraries interact with PROPATH during a Progress session:

- The first time you run a member procedure from a standard or memory-mapped library that is specified in the PROPATH, Progress locates the procedure by searching through each directory and library in the PROPATH until it finds the procedure. To search a library for a member procedure, Progress must open the library. When Progress opens a standard library, it loads the procedure into local memory. When Progress opens a memory-mapped library, it maps the library in shared memory.

- When searching through directories and libraries in the PROPATH, Progress starts at the beginning of the PROPATH and searches each directory and library, in defined order, until it finds the procedure. Thus, placing the library at the beginning of the PROPATH improves performance.

- A library remains open until the end of your Progress session or until you remove the library from the PROPATH. If you remove a library from the PROPATH while a member procedure is still active (either running, or waiting for a subprocedure to return), Progress displays an error message and the library remains open until you end your Progress session. Otherwise, Progress closes a standard library and unmaps a memory-mapped library.

- If you use a SEARCH or RUN statement to open a library that is not in the PROPATH, the library remains open until you end your Progress session.
- If you use libraries, Progress accesses r-code files as if you had specified the Quick Request (–q) startup parameter. That is, Progress searches the PROPATH only on the first reference to a procedure. Thereafter, if the procedure still resides in memory, in the local session compile file, or in an r-code library, Progress reuses the procedure instead of searching the PROPATH again. For more information about the Quick Request (–q) startup parameter, see the *Progress Startup Command and Parameter Reference*.

B.4 R-code Execution

Progress executes r-code procedures in different ways, depending on whether you store the r-code files in an r-code library and the type of library.

When executing an r-code procedure from an operating system file in a directory, or from a standard r-code library, Progress accesses and executes the r-code file segments in an execution buffer in local memory. For more information about the standard r-code execution environment, see the "Standard R-code Execution Environment" section later in this appendix.

When executing an r-code procedure from a memory-mapped r-code library, Progress accesses and executes the r-code file segments in shared memory. For more information about the memory-mapped r-code execution environment, see the "Memory-mapped R-code Execution Environment" section later in this appendix.

For information about monitoring execution environment activity during a Progress client session, see the "R-code Execution Environment Statistics" section later in this appendix.

For information about monitoring and optimizing r-code performance, see the *Progress Client Deployment Guide*.

B.4.1 Standard R-code Execution Environment

At run-time, Progress manages the execution of a standard r-code procedure from either an operating system file or a standard procedure library using the following components:

- **Execution buffer** — The portion of local memory that Progress allocates and uses to store the chain of loaded r-code segments for all procedures executing in local memory.
- **Session sort file (.srt)** — A file that Progress uses to dynamically swap r-code segments in and out of the execution buffer.

- **Segment descriptor table** — An in-memory table that references the r-code segments required by all executing procedures, including the location of each r-code segment in the execution buffer and usage count information.

- **R-code directory** — An in-memory table that contains information about each executing r-code procedure, including r-code size, usage count, segment descriptions, and a reference to the segment descriptor table for each segment.

Progress uses the segment descriptor table and the r-code directory to manage r-code procedures from operating system files, standard libraries, and memory-mapped libraries.

Figure B–2 shows the layout for the standard r-code execution environment.

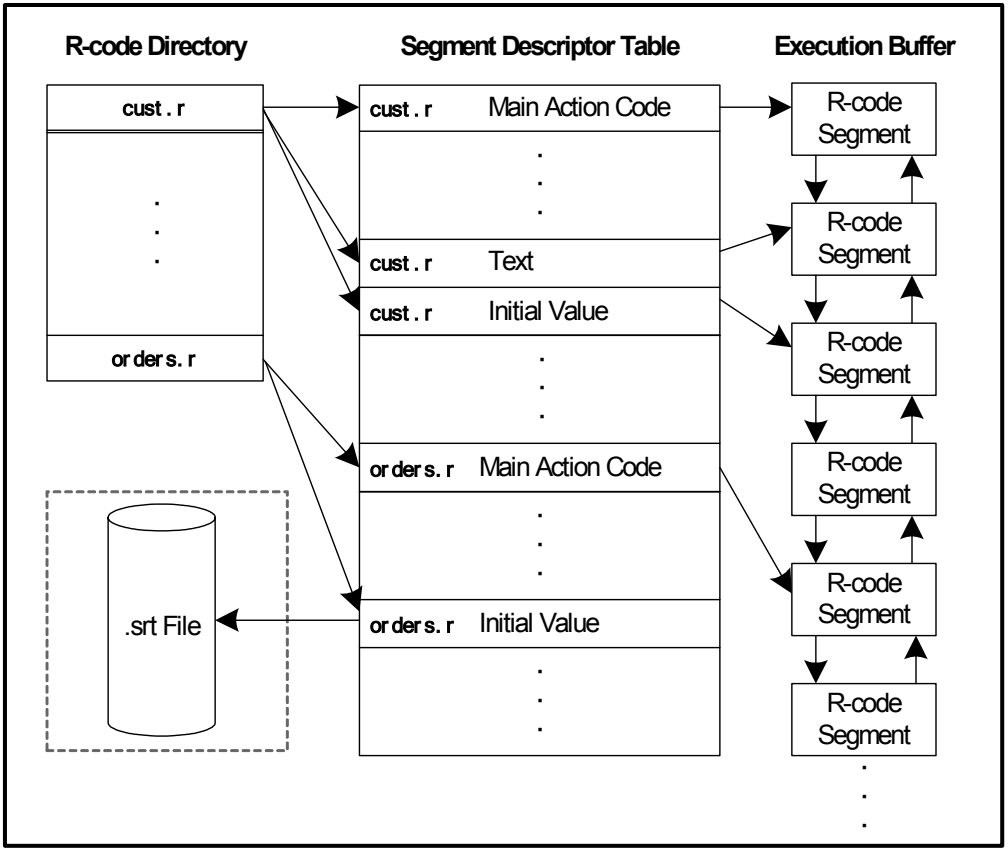


Figure B–2: Standard R-code Execution Environment

In Figure B–2, Progress located and loaded the `cust.r` and `orders.r` files into local memory from either operating system files or standard libraries. The execution buffer is shown with three `cust.r` segments and two `orders.r` segments. Note that one `orders.r` segment is located in the execution buffer, while the other segment is swapped to the session sort file. When space in the execution buffer is needed for new r-code segments, Progress uses the session sort file to swap out the least-recently used segments. When Progress needs a segment that has been swapped to the session sort file, it reloads the segment from the session sort file into the execution buffer.

Standard Execution Sequence

When you run a standard r-code procedure for the first time, from either an operating system file or a standard library, Progress loads and executes the procedure as follows:

1. Opens the procedure file (.p or .r) or procedure library (.pl), if the library is not already open.

2. Reads the r-code procedure into memory and creates an r-code directory entry for the procedure. Progress first compiles a procedure into r-code, if necessary.

3. Registers each required r-code segment in the execution environment as follows:

   a. Loads the r-code segment at the head of the segment chain in the execution buffer

   b. Adds an r-code segment entry to the segment descriptor table that references the segment in the execution buffer

   c. Inserts a segment descriptor reference in the r-code directory entry for the procedure

   If all the required r-code segments do not fit in the execution buffer, Progress attempts to free space by swapping r-code segments already in the buffer to the session sort file. If Progress cannot free enough space by swapping segments, it increases the execution buffer ceiling and allocates more space for the execution buffer.

   NOTE: When accessing r-code procedures stored in a standard library, Progress does not swap r-code segments to the session sort file unless you specify the PROLIB Swap (–pls) startup parameter. By default, if Progress needs an r-code segment in a standard library, it reloads the segment into the execution buffer from the library in local memory.

4. Once the required r-code segments in the procedure are registered in the execution environment, the interpreter begins executing the r-code procedure at the start of the first main action code segment and accesses the remaining segments directly from local memory as required.

Standard Execution Environment Limits

The number of standard r-code procedures that you can run at one time and the memory used are determined by the following factors:

- **R-code directory size** — The default is 100 entries, the minimum is 5 entries, and the maximum is 500. You can set the initial number of entries using the Directory Size (–D) startup parameter. Progress dynamically increases the directory size up to the maximum, as required. Use the Hardlimit (–hardlimit) startup parameter to force Progress to adhere to the limit specified by the Directory Size (–D) startup parameter.

- **Execution buffer ceiling** — The default is 3096K. You can set the initial ceiling for this buffer up to 65,534K using the Maximum Memory (–mmax) startup parameter. Progress dynamically increases the execution buffer size up to the maximum, as required. Use the Hardlimit (–hardlimit) startup parameter to force Progress to adhere to the limit specified by the Maximum Memory (–mmax) startup parameter.

- **Available memory** — Available memory is a factor only if it is smaller than the execution buffer ceiling or Progress needs to allocate memory beyond that ceiling.

Standard R-code Segment Management

While Progress loads an r-code file, all of its segments are locked in memory. After all required segments are loaded for the procedure, Progress unlocks all segments except its main action code segments and text segment. These segments stay locked in memory until execution of the r-code file terminates. Internal procedure action code segments stay locked only until they return to the invoking procedure and are relocked each time they execute.

When a standard r-code segment does not fit in the execution buffer, Progress attempts to free space by swapping r-code segments already in the buffer to the session sort file. Progress can swap out any unlocked segments. Progress removes these segments from the tail end of the execution buffer chain, in least recently used (LRU) order.

If Progress cannot free enough memory for a newly loaded segment by swapping out older segments, it dynamically increases the maximum execution buffer size (execution buffer ceiling) and allocates the required memory up to the memory available.

When Progress needs a segment that has been swapped to the sort file, it reloads the segment from the sort file into the execution buffer. However, Progress keeps the reloaded segments in the sort file.

NOTE: When accessing r-code procedures stored in a standard library, Progress does not swap r-code segments to the session sort file unless you specify the PROLIB Swap (–pls) startup parameter. By default, if Progress needs an r-code segment in a standard library, it reloads the segment into the execution buffer from the library in local memory.

B.4.2 Memory-mapped R-code Execution Environment

At run time, Progress manages the execution of memory-mapped r-code procedures using the following components:

- **Shared memory buffer** — The portion of shared memory that the operating system allocates and uses to store r-code segments for procedures in a memory-mapped procedure library.
- **Segment descriptor table** — An in-memory table that references the r-code segments required by all executing procedures, including the location of each r-code segment in memory and usage count information.
- **R-code directory** — An in-memory table that contains information about each executing r-code procedure, including r-code size, usage count, segment descriptions, and a reference to the segment descriptor table for each segment.

Progress uses the segment descriptor table and the r-code directory to manage r-code procedures from operating system files, standard libraries, and memory-mapped libraries.

Figure B–3 shows the layout for the memory-mapped r-code execution environment.

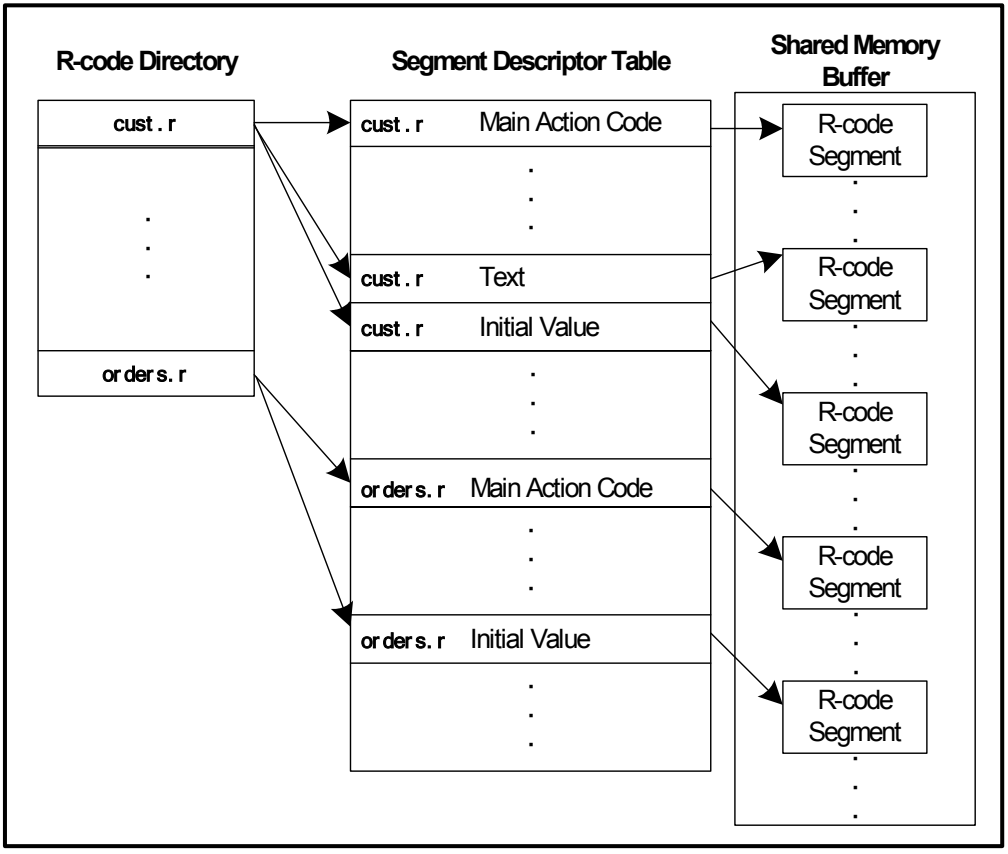


Figure B–3: Memory-mapped R-code Execution Environment

In Figure B–3, Progress located the `cust.r` and `orders.r` files in a memory-mapped library. Note that all r-code segments in a memory-mapped library are located in shared memory. When Progress needs a segment, it executes the segment directly from shared memory.

Memory-Mapped Execution Sequence

When you run a memory-mapped r-code procedure for the first time, Progress loads and executes the procedure as follows:

1. Opens the procedure library, if not already open, and memory-maps the library in shared memory.

2. Reads the r-code procedure and creates an r-code directory entry for the procedure.

3. Registers each required r-code segment in the execution environment as follows:

   a. Adds an r-code segment entry to the segment descriptor table that references the segment in shared memory

   b. Inserts a segment descriptor reference in the r-code directory entry for the procedure

4. Once the required r-code segments in the procedure are registered in the execution environment, the interpreter begins executing the r-code procedure at the start of the first main action code segment and accesses the remaining segments directly from shared memory as required.

Memory-Mapped Execution Environment Limits

The number of memory-mapped r-code procedures that you can run at one time and the shared memory used are determined by the following factors:

- **R-code directory size** — The default is 100 entries, the minimum is 5 entries, and the maximum is 500. You can set the initial number of entries using the Directory Size (–D) startup parameter. Progress dynamically increases the directory size up to the maximum, as required. Use the Hardlimit (–hardlimit) startup parameter to force Progress to adhere to the limit specified by the Directory Size (–D) startup parameter.

- **Available memory** — The number of memory-mapped r-code libraries you can have open is limited by the amount of shared memory available on the system.

Memory-Mapped R-code Segment Management

When Progress needs a memory-mapped r-code segment, it executes the segment directly from shared memory. Progress does not store active memory-mapped r-code segments in the execution buffer. Nor does it swap non-active segments to the session sort file. Progress relies on the operating system to manage the swapping of r-code segments in and out of shared memory.

B.4.3 R-code Directory Management

If the r-code directory is full when loading a new standard or memory-mapped r-code file, Progress can reuse existing directory entries for any r-code files no longer in use.

To reuse an r-code directory entry, Progress performs the following steps:

1. Identifies the least recently used (LRU) r-code file.

2. Frees the r-code directory entry for the LRU r-code file.

3. Frees all of the segment descriptor entries in the segment descriptor table for the LRU r-code file.

4. When reusing a standard r-code directory entry, Progress removes all r-code segments for the LRU r-code file from both the execution buffer and the session sort file. When reusing a memory-mapped r-code directory entry, the memory-mapped r-code file remains mapped in shared memory.

When Progress needs to reload a standard r-code file, it follows the standard execution sequence. When Progress needs to reload a memory-mapped r-code file, it follows the memory-mapped execution sequence.

B.4.4 R-code Execution Environment Statistics

You can monitor execution environment activity using the Statistics (–y) and Segment Statistics (–yd) startup parameters. These parameters cause Progress to write memory statistics to the `client.mon` file. The –yd parameter provides all of the information about memory usage available with –y plus additional information about r-code segments loaded during a client session. For more information about the –yd and –y startup parameters, see the *Progress Startup Command and Parameter Reference*.

For information about monitoring and optimizing r-code performance, see the *Progress Client Deployment Guide*.

Figure B–4 through Figure B–8 show sections of `client.mon` output generated with the –yd startup parameter.

In Figure B–4, the "Progress client startup options" section shows the r-code directory size (–D), in this case, set to the default of 100 entries. The –y startup parameter also generates this information.

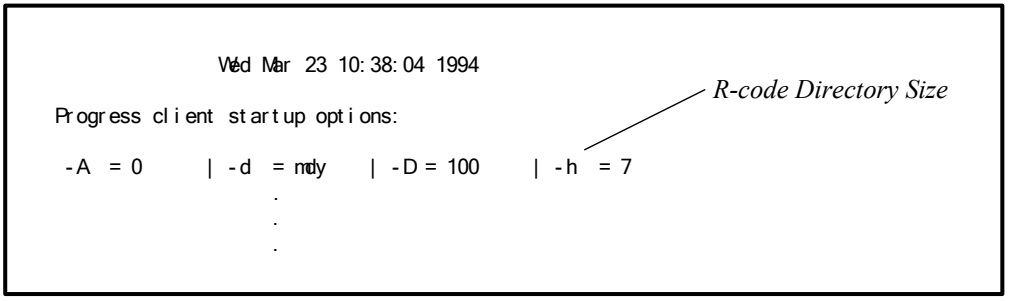


Figure B–4: Progress Client Startup Options For –yd

In Figure B–5, the "Execution buffer map" section shows the order and size of procedures loaded into the execution buffer.

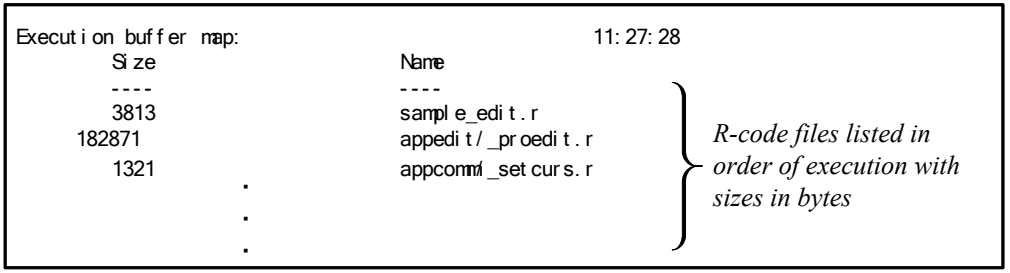


Figure B–5: Execution Buffer Map For –yd

In Figure B–6, the "Per procedure temp file access statistics" section lists each r-code segment that has been read or written to the session sort file (.srt). Each segment is listed under its r-code file by segment type ("Int–Proc Action"), number (5), and size (1364 bytes).

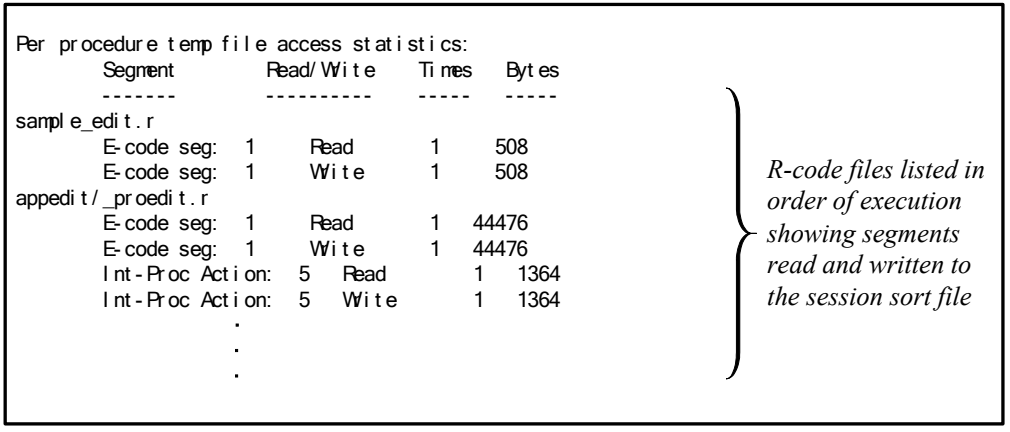


Figure B–6: Accessing the Session Sort File For –yd

In this example, all segments shown have been read and written once. A large number of segments read or written a large number of times indicates a likely need for more memory.

In Figure B–7, the "Per procedure segment information" section lists all the r-code segments loaded during the session. Each segment is listed under its r-code file by segment type ("Int–Proc"), number (2), and size (816 bytes).

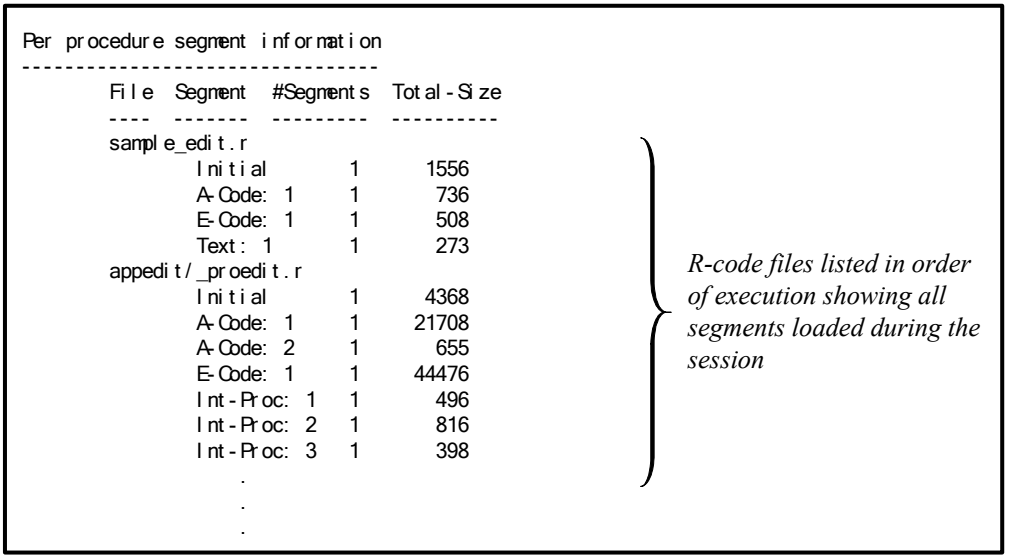


Figure B–7: Procedure Segment Information For –yd (Part 1)

Thus, if there are three internal procedure action segments in `appedit/_proedit.r`, they are listed in order (Int–Proc: 1 through 3). Likewise, multiple main action code segments are listed in order (A–Code: 1 and 2). The number of segments for each entry is always 1. Note that the "Initial" segment has no segment number, because there is never more than one of this segment type per r-code file.

In Figure B–8, the listed sections provide memory and segment usage summaries. The –y startup parameter also generates this information.

```
----------------------------------------
Program access statistics:    Times    Bytes
                     .
                     .
                     .

Memory usage summary:       Current    Max Used       Limit (Bytes)
Stack usage (-s):               60        7248       40960
Local buffer usage:            864       14704
R-code Execution Buffer:     530025      710609      710656

Segment Descriptors Usage: (numbers)
Max Used:      508      Limit:      720

-----------------------------------------------
```

Figure B–8: Procedure Segment Information For –yd (Part 2)

The "R-code Execution Buffer" statistics show how far your application pushes the execution buffer ceiling. The "Segment Descriptors Usage" statistics shows how close your application is to running out of segment descriptors, and thus, whether you need to optimize the number of r-code segments in your application.

B.5 R-code Portability

R-code is portable between any two environments if the following are true:

- The user interface is the same — graphics mode or character mode.

  NOTE: If a procedure contains no user interface code, its r-code can run across user interfaces without change. This is typical, for example, of database schema trigger procedures.

- The DataServer or combination of DataServers connected during compilation is the same.

- The machine classes are compatible.

All machines that require four-byte alignment or less are compatible. Thus, r-code compiled on one of these machines (for example, a Sun SPARCstation) can run on most other machines. One of the few machines not compatible with this class is the AS/400, which requires greater than four-byte alignment.

Thus, you can use the same r-code compiled for Windows on any DataServer-compatible Windows system, and you can use the same r-code compiled in character mode on any DataServer-compatible character-mode system.

B.6 Code Page Compatibility

The Internal Code Page (-cpinternal) startup parameter determines the internal code page that Progress uses in memory. The code page in which r-code is compiled and the internal code page of the session in which the r-code runs must be compatible. That is, the r-code code page must be either the internal code page, undefined, or convertible to the internal code page.

If the code page for a standard r-code procedure, from either an operating system file or a standard library, is not the same as the session's internal code page, Progress performs an in-memory conversion of the r-code text segments to the session's internal code page.

NOTE: If you use the R-code In Code Page (–cprcodein) startup parameter to specify a code page for reading r-code text segments, Progress reads the text segments as if they were written in that code page, even if the text segments were written in a different code page.

Since r-code procedures in memory-mapped procedure libraries are read-only, Progress cannot perform an in-memory code page conversion of r-code text segments. If the code page of a memory-mapped r-code procedure is not compatible, Progress will not execute the r-code. It will report an error and stop execution.

For more information about the Internal Code Page (–cpinternal) and R-code In Code Page (–cprcodein) startup parameters, see the *Progress Startup Command and Parameter Reference*.

B.7 Database CRCs and Time Stamps

At run time, Progress must ensure that r-code files and the database(s) that they access are compatible. Otherwise, if the schema has changed since the code was compiled, running the code might damage the database. Progress supports two techniques for checking this compatibility—time stamp validation and cyclic redundancy check (CRC) validation. Each technique prevents you from running unauthorized or data-incompatible procedures against a particular database environment.

B.7.1 Time Stamp Validation

For each table, Progress maintains a time stamp indicating when the schema was last changed. When compiling a procedure, Progress inserts the time stamp into the r-code for each database table accessed by the procedure. This time stamp is the value of the _Last–change field in the metaschema _File record for each table. Note that sequence changes have no effect on time stamp validation.

R-code Execution With Time Stamps

When executing a procedure using time stamp validation, Progress checks to make sure that the time stamps for all accessed tables match those contained in the r-code. If a time stamp does not match, the procedure does not execute.

To use time stamp validation, compile the procedure with each database connected using the Time Stamp (–tstamp) connection parameter. Otherwise, Progress uses CRC validation.

NOTE: The Time Stamp (–tstamp) parameter is obsolete, and is supported only for backward compatibility.

B.7.2 CRC Validation

For each table, Progress computes a CRC value (*database* CRC) from selected elements of the table metaschema. When compiling a procedure, Progress inserts the database CRC for each database table the procedure accesses. The database CRC for each table is stored in the metaschema _File record _CRC field for the table. A matching CRC ensures that the schema referenced by the r-code is compatible with the table schema, regardless of its time stamp.

Time Stamps and CRCs

Time stamps and CRCs both help maintain database integrity, but they do it differently. Time stamps change whenever a table schema is updated or recreated at another time or place; CRCs change only when certain schema elements critical to field or record definitions change. As long as a table is structurally compatible with the r-code that references it, the CRCs match and the r-code can execute. It does not matter when or where the table's schema was created or updated.

R-code Execution With CRCs

When executing a procedure using CRC validation, Progress follows these steps:

1. For each table accessed by the r-code, check the time stamp. If it matches, execute the procedure. Otherwise, go to Step 2.

2. If the table is in a database that was not connected using the Time Stamp (–tstamp) parameter when the procedure was compiled, check the CRC. If it matches, execute the procedure. Otherwise, reject the procedure.

Thus, if the time stamps match, Progress assumes that the database schema and r-code must be compatible. Otherwise, the CRCs determine the outcome.

NOTE: The Time Stamp (–tstamp) parameter is obsolete, and is supported only for backward compatibility.

B.7.3 CRC Calculation

Table B–2 lists the metaschema fields that are involved in each CRC calculation.

Table B–2: Metaschema Fields In CRC Calculation

Metaschema Table			
_File	**_Field**	**_Index**	**_Index–Field**
_File–Name _DB–lang	_Field–Name _Data–Type _dtype _sys–field _field–rpos _Decimals _Order _Extent _Fld–stdtype _Fld–stlen _Fld–stoff _Fld–case	_Index–Name _Unique _num–comp	(_Field–Name)[1] _Ascending _Abbreviate _Unsorted

[1] The _Field–Name for index fields is referenced in the _Field table using _Field–recid in _Index–Field.

CRC Calculation For Tables

All of the listed metaschema fields, participate in the CRC calculation for each database table. The resulting CRC is stored in the _CRC field of the _File record for each table. A change to any of these fields automatically changes the value of _CRC.

B.7.4 CRC Versus Time Stamp Validation

The main advantage of using CRC validation is that you can run the same r-code against different databases as long as the databases have sufficiently similar schemas. This allows you to deploy code and database updates while avoiding either of the following effects:

- Forcing the user to dump and reload their data into a new database that you supply with new r-code
- Forcing the user to compile new source code (possibly encrypted) that you supply with new data definitions to update their database schema

Depending on the schema changes, the user still might have to dump and reload some data, but you can always supply r-code for their updated database.

Database Changes That Affect Time Stamps

You must recompile procedures that reference a database table in which you have made any of the following changes:

- Deleted the table
- Changed the table name
- Changed the database language or authorization key
- Added or deleted a field in the table
- Changed the name or order of a field
- Added or deleted an index
- Modified an index by changing its name, uniqueness, field components, collation order, or the abbreviation option

Although you do not have to recompile procedures when you make other schema changes, such changes do not apply to procedures that you do not recompile. For example, changes to the following features appear only when you recompile:

- Field appearance — format, label, column label, and decimal
- Data entry — initial value, mandatory value, case sensitivity, validation expression, delete validation, and security
- User information — help messages, validation messages, and description

Database Changes That Affect CRCs

You must recompile procedures using CRC-based deployment for the same schema changes that change a time stamp. In addition, you must recompile procedures that reference a database table in which you have made these changes to a field:

- The data type or extent (array size)
- The number of decimals
- The case sensitivity

Table B–2 lists all the metaschema fields involved in database CRC calculation.

NOTE: The order in which indexes are defined has no effect on database CRCs.

Time Stamp-based Deployment

Using time stamp validation, if you deploy r-code, you must compile the procedures against a schema-updated version of the user's database and send both the r-code and the new database to the user. Otherwise, the r-code cannot run, and you must deploy source code for the user to compile along with new database definitions to update their schema.

CRC-based Deployment

Using CRC validation, if you deploy r-code, you only have to compile the procedures against a database with the same basic structure as the end-user database. It does not matter when or where the database was created. If you make schema changes to an existing database, you can distribute the newly compiled r-code to end users, along with a data definition files to install the schema changes.

The flexibility of CRC-based deployment makes it easier to distribute a new version of an existing application. To deploy an application update based on CRC validation, follow these steps:

1 ♦ Install a copy of the production database from the target site in your development environment.

2 ♦ Create an incremental data definition file that captures the difference between your development database and the production copy. For information about creating an incremental data definition file, see the *Progress Database Administration Guide and Reference*.

3 ♦ Apply the incremental data definition file to your copy of the production database.

4 ♦ Recompile all procedures affected by the schema changes against your copy of the production database.

5 ♦ Apply the incremental data definition file to the original production database (on site). The CRC for the original production database should now be identical to the production copy, but probably not identical to your development database.

6 ♦ Distribute the new r-code files (from Step 4) to the updated production site.

The steps for CRC-based deployment are essentially the same as those for time stamp-based deployment, but you gain these advantages:

- You are not required to encrypt and give end users source code.
- End users do not have to compile your source code.
- You do not have to send a copy of a new database to end users.
- End users do not have to dump and reload all their data into your new database.

Differences In R-code Access Authorization

One possible drawback of using CRC validation is that anyone having a Progress development environment can create a counterfeit database (a database with the same structure as your database). The user can then write a program, compile it, and run it against your database. Using time stamp validation prevents this because the time stamps in the counterfeit code do not match those in your database.

However, you can also combine CRC validation with additional security to prevent unauthorized r-code access. The DBAUTHKEY option of the PROUTIL utility allows you to set an authorization key for your database. When you compile your source code, Progress includes the value of this authorization key in your r-code. You can also use the RCODEKEY option of PROUTIL to insert the authorization key in existing r-code. Any r-code that does not include the correct authorization key cannot run against your database. For more information about using PROUTIL to create and set database authorization keys, see the *Progress Database Administration Guide and Reference*.

CRC and Time Stamp Availability

During compilation, Progress always inserts both database time stamps and CRCs in the r-code. If a database is not connected with the Time Stamp (–tstamp) parameter during procedure compilation, Progress turns on a bit in the r-code. This bit tells the run-time interpreter to check the CRC if the time stamp check fails.

NOTE: The Time Stamp (–tstamp) parameter is obsolete, and is supported only for backward compatibility.

B.8 R-code CRCs and Procedure Integrity

When Progress compiles a procedure, it calculates a special CRC based on the procedure name and code content, and stores it in the r-code. This *r-code CRC* (as distinguished from database CRC) is useful to ensure the integrity and security of Progress procedures, especially for schema triggers.

If you choose to have CRCs checked for a schema trigger, the r-code CRC of the trigger procedure definition is stored in the trigger schema. The trigger procedure cannot run unless it has a matching r-code CRC. This prevents a trigger procedure with the same name, but different 4GL code, from being improperly substituted for the original trigger procedure. Such a substitution can cause damage to your database or override your security.

For other application r-code files, you can use the RCODE–INFO handle to build a procedure security table that contains the name and r-code CRC of each r-code file in your application (except the startup procedure). Before running a procedure, your application can use the RCODE–INFO handle to validate the r-code CRC of its r-code file against the entry established for it in the procedure security table.

NOTE: If there is no r-code for the procedure, any source (.p) procedure returns an unknown (?) CRC value.

B.8.1 Assigning CRCs To Schema Triggers

When you create a schema trigger in the Data Dictionary, you can specify CRC checking. Then, when you save the trigger procedure, the Data Dictionary calculates its r-code CRC and stores it in the trigger metaschema record (_File–Trig for table triggers and _Field–Trig for field triggers).

If you want CRC checking for all your schema triggers, you must specify it for each schema trigger in your database. For more information, see the *Progress Basic Development Tools* manual (Character only) and, in graphical interfaces, the on-line help for the Data Dictionary.

B.8.2 Validating CRCs For Schema Triggers

When a database event occurs in your application (such as creating a record or assigning a field), Progress checks whether the CRC of the corresponding trigger procedure matches the CRC in the trigger metaschema record. If it matches, Progress executes the procedure. Otherwise, Progress returns a run-time error.

Trigger CRC validation works for both r-code and source versions of a trigger procedure. For source versions, Progress calculates the CRC during the session compile.

B.8.3 RCODE–INFO Handle

The RCODE–INFO handle allows you to read the r-code CRC from an r-code file. You can use this value to build a procedure security table, and then to verify the integrity of your application r-code against it. You can also use this value in a deployment procedure that you run in a secure context to automatically install database schema trigger definitions.

Getting the R-code CRC

To read the r-code CRC from an r-code file, first set the FILE–NAME attribute of the RCODE–INFO handle to the pathname of your r-code file. Then read the value of the CRC–VALUE attribute:

```
DEFINE VARIABLE rCRC AS INTEGER.
RCODE-INFO:FILE-NAME = "sports/crcust.r".
rCRC = RCODE-INFO:CRC-VALUE.
```

NOTE: The RCODE–INFO handle cannot read the r-code CRC from a session compile. For source (.p) procedures, the CRC–VALUE attribute returns the unknown value (?).

Example — Verifying Application R-code Integrity

This example shows how you might provide r-code integrity and security in your application.

1 ♦ Build the procedure security table, RcodeSecurity, with these fields:

- Filename — CHARACTER field for the pathname of each r-code file.
- CRC — INTEGER field for the r-code CRC of the specified r-code file.

2 ♦ Construct a text file, `crctable.dat`, that contains a list of the relative pathnames for all the secure procedures called by your application.

3 ♦ Compile and save the r-code for all the secure procedures in your application.

4 ♦ Run a procedure that contains code like this to build the RcodeSecurity table:

```
DEFINE VARIABLE i AS INTEGER.
DEFINE VARIABLE proc-name AS CHARACTER FORMAT "x(32)".
DEFINE VARIABLE rCRC AS INTEGER.

INPUT FROM "crctable.dat". /* List of r-code file pathnames */

i = 0.
REPEAT:
    SET proc-name.
    FIND RcodeSecurity WHERE Filename = proc-name NO-ERROR.
    IF NOT AVAILABLE(RcodeSecurity) THEN
        CREATE RcodeSecurity.
    RCODE-INFO:FILE-NAME = proc-name.
    SET RcodeSecurity.Filename = proc-name
        RcodeSecurity.Crc = RCODE-INFO:CRC-VALUE.
    i = i + 1.
END.

INPUT CLOSE.
MESSAGE i "procedure security records created".
```

5 ♦ At each point where you call a secure procedure in your application, insert this code:

```
FIND RcodeSecurity WHERE Filename = "secret.r".
RCODE-INFO:FILE-NAME = "secret.r".

IF RcodeSecurity.Crc = RCODE-INFO:CRC-VALUE THEN
    RUN secret.
ELSE DO:
    MESSAGE "Procedure secret.r is invalid.".
    QUIT.
END.
```

Example — Deploying Schema Triggers

The following procedure installs new schema trigger definitions in an existing database. It reads dump files that contain the new data for table and field metaschema trigger records and updates these records with the new trigger procedure names and CRCs:

p-trload.p *(1 of 2)*

```
/* p-trload.p */
DEFINE VARIABLE i AS INTEGER.
DEFINE VARIABLE event AS CHARACTER FORMAT "x(6)".
DEFINE VARIABLE proc-name AS CHARACTER FORMAT "x(32)".
DEFINE VARIABLE table-name AS CHARACTER FORMAT "x(32)".
DEFINE VARIABLE field-name AS CHARACTER FORMAT "x(32)".
DEFINE VARIABLE rCRC AS INTEGER.
INPUT FROM "_file-tr.dat". /* Table trigger data */
i = 0.
_fl-loop:
REPEAT:
  SET table-name event proc-name.
  RCODE-INFO:FILE-NAME = proc-name.
  rCRC = RCODE-INFO:CRC-VALUE.
  FIND _file WHERE _file-name = table-name.
  FIND _file-trig WHERE _file-recid = RECID(_file) AND
                        _event = event NO-ERROR.
  IF AVAILABLE _file-trig THEN DO:
    IF  _File-trig._Proc-name = proc-name
    AND _File-trig._Trig-CRC  = rCRC THEN
        NEXT _fl-loop.
    ELSE DO:
    /* Progress doesn't let you modify a trigger record, so delete and
       recreate. */
      DELETE _File-trig.
      CREATE _File-trig.
      ASSIGN _File-trig._File-recid = RECID(_File)
             _File-trig._Event      = event
             _File-trig._Override = TRUE
             _File-trig._Proc-Name  = proc-Name
             _File-trig._Trig-CRC   = rCRC
             i = i + 1.
    END.
  END.
END.
```

p-trload.p *(2 of 2)*

```
INPUT CLOSE.
MESSAGE i "_file-trig records updated.".
INPUT FROM "_field-t.dat". /* Field trigger data */
i = 0.
_fld-loop:
REPEAT:
  SET table-name field-name event proc-name.
  RCODE-INFO:FILE-NAME = proc-name.
  rCRC = RCODE-INFO:CRC-VALUE.
  FIND _file WHERE _file-name = table-name.
  FIND _field WHERE _file-recid = RECID(_file) AND
                    _field-name = field-name.
  FIND _field-trig WHERE _field-trig._file-recid = RECID(_file) AND
                         _field-trig._field-recid = RECID(_field) AND
                         _event = event NO-ERROR.
  IF AVAILABLE _Field-trig
      AND _Field-trig._Proc-name = proc-name
      AND _Field-trig._Trig-CRC  = rCRC THEN NEXT _fld-loop.
  ELSE DO:
    DELETE _Field-trig.
CREATE _Field-trig.
    ASSIGN
      _Field-trig._File-recid  = RECID(_File)
      _Field-trig._Field-recid = RECID(_Field)
      _Field-trig._Event       = event
      _Field-trig._Override    = TRUE
      _Field-trig._Proc-Name   = proc-Name
      _Field-trig._Trig-CRC    = rCRC
      i = i + 1.
  END.
END.
INPUT CLOSE.
MESSAGE i "_field-trig records updated.".
```

NOTE: As this example shows, you must use the RECID function rather than the ROWID function to reference metaschema table record locations.

C

AppServer Internet Adapter

The AppServer Internet Adapter (AIA) and the Secure AppServer Internet Adapter (AIA/S) Web-enable the AppServer by supporting HTTP and HTTPS protocols for sending information across the Internet. In addition, the HTTP and HTTPS protocols provide a way for AppServer clients to access an AppServer when the client and AppServer are separated by firewalls that limit connections to HTTP and HTTPS.

This document contains information about:

- AppServer Internet Adapter Architecture
- Installing and Licensing the AppServer Internet Adapter
- Installing and Configuring Web Servers and Java Servlet Engines
- Administering and Configuring an AppServer Internet Adapter
- Viewing AppServer Internet Adapter Connection and Configuration Information

In addition to reading this document, it is essential that you read the *Building Distributed Applications Using the Progress AppServer* manual before you develop your Web-enabled applications for use with AIA. You can also use the Progress Knowledge Base on the Web (`http://www.Progress.com/support`) for information about Web-enabled 4GL clients. For Open Client applications, you should read the *Progress Open Client Developer's Guide*.

C.1 AppServer Internet Adapter Architecture

Running AppServer applications over the Internet can present challenges due to the restrictive access policies enforced by many company networks. Companies often use firewalls to protect their internal systems from unauthorized access. A firewall is a computer and software that sits between a private network and the Internet. The firewall restricts the types of protocols that can travel into the internal network. The firewall often restricts the ports these protocols can use. For example, it is common to limit traffic originating outside a firewall to those applications using HTTP on port 80.

AIA is a Java servlet that is run by a Java servlet engine (JSE) and allows Java programs to run on a Web server. AIA supports deployment of distributed applications over the Web by extending the AppServer architecture to support the HTTP and HTTPS protocols. When you use AIA, communication between an AppServer client and an AppServer is encapsulated within standard HTTP requests and "tunnels" those requests through a Web server. AIA then converts requests that are encapsulated within HTTP or HTTPS to the standard AppServer protocol. This conversion process allows AppServer clients to communicate through firewalls without requiring a network administrator to allow additional protocols or ports to be accessed through a firewall.

HTTPS is HTTP on a Secure Socket Layer (SSL) connection rather than the standard unencrypted TCP/IP connection. HTTPS is used when privacy, integrity, and server authentication are needed in addition to HTTP tunneling. To use the HTTPS protocol, you can use any Progress Web-enabled client with the Client-Side Security package.

Figure C–1 illustrates the AIA and AIA/S architecture.

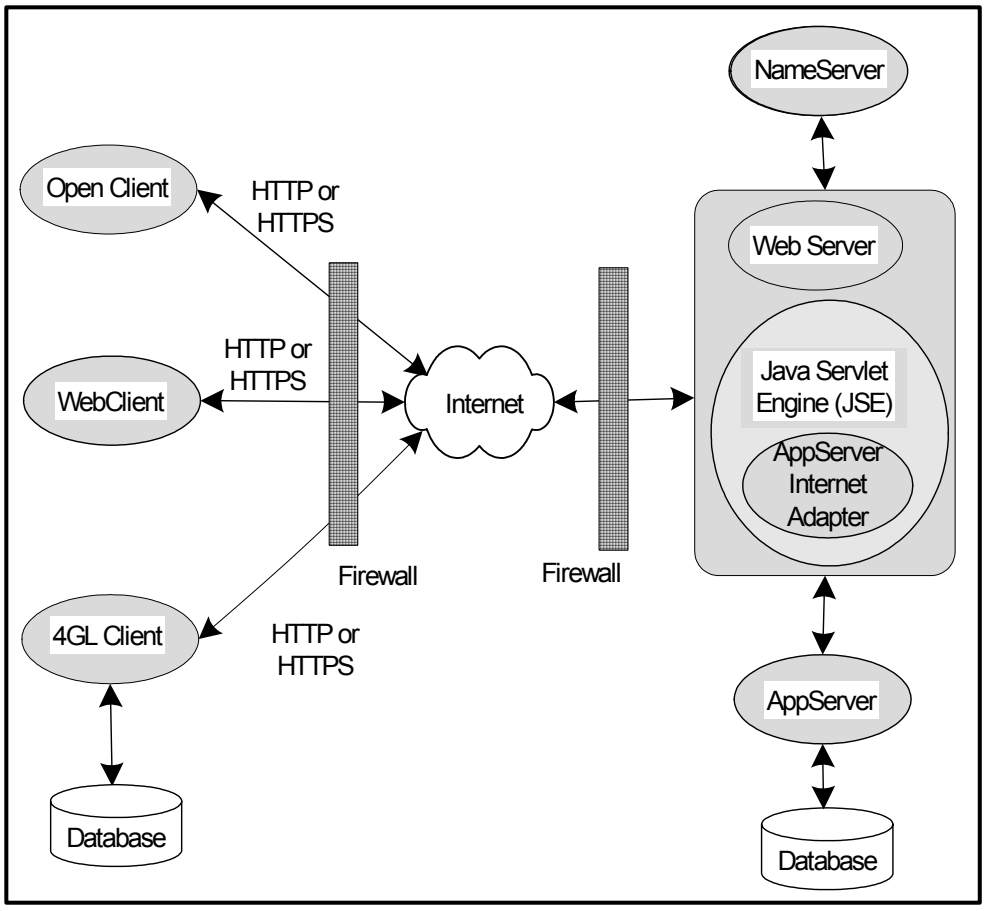


Figure C–1: AppServer Internet Adapter Architecture

C.1.1 Using the AppServer Internet Adapter With HTTP

When using HTTP, all requests between the client and the AppServer go through a Web server that forwards each request to the AppServer. Depending on the client type, an AppServer client indicates that Progress should use HTTP tunneling as follows:

- For a 4GL client, including WebClient, the client uses the `-URL` connection parameters of the SERVER handle by specifying a URL with an HTTP protocol.
- For an Open Client, you specify the URL in the AppObject Proxy object that you can create with `ProxyGen`.

The URL contains the host and port of the Web server, the path to the AIA instance, as well as the requested application service. For information about how a client connects to an AppServer using HTTP see the *Progress Language Reference* for 4GL Clients or the *Progress Open Client Developer's Guide* for Open Clients.

Each AIA instance can either connect directly to an AppServer or use a controlling NameServer to connect. How the connection is made is specified when you configure the AIA instance. Each request sent from an AppServer client to the Web server is encapsulated within one or more standard HTTP POST requests. The URL contained within the POST request indicates to the Web server that it should forward the request to AIA.

When the AIA instance receives the connection request from an AppServer client, the requested application service name is included along with the *`userid`*, *`password`*, and *`appserver-info`* values that are normally passed in a connection request to an AppServer. AIA then passes the application service name either to the controlling NameServer and the Name Server passes the request to the AppServer, or AIA passes the request directly to the AppServer. For more information, see the "Administering and Configuring an AppServer Internet Adapter" section.

NOTE: Because the HTTP protocol is connectionless, you cannot use the termination of the connection with the AppServer client as an indication that the AppServer client's connection to the AppServer should be terminated. Due to failures at the client site, as well as the network between the client site and the Web server, it is possible that an AppServer client could connect to an AppServer and never gracefully disconnect. To guard against the possibility of an AIA instance maintaining orphaned connections, you can use the `idleConnectionTimeout` property in the `ubroker.properties` file to configure an AIA instance to automatically terminate the connection if a connection is idle for more than the specified time. For more information, see the "Administering and Configuring an AppServer Internet Adapter" section.

Supported Client Platforms

For the following clients, HTTP is supported on these platforms:

- WebClient is supported on the following Windows platforms: 98, NT 4.0, ME, 2000, and XP
- 4GL client is supported on all platforms on which Progress currently runs
- Open client is supported on all platforms that have Microsoft or Javasoft compatible JVMs

Supported Server Platforms

On server machines, you can use any Web server that supports HTTP. For more information, see the "Installing and Configuring Web Servers and Java Servlet Engines" section.

C.1.2 Using the AppServer Internet Adapter With HTTPS

HTTPS allows Web-enabled clients with the Client-Side Security package to connect to a Secure AppServer over the Internet. Depending on the type of client you use, Client-side security is provided as follows:

- **WebClient** — This installation includes the Client-Side Security package.
- **4GL Client** — This installation does not include the Client-Side Security package. You must install the Client-Side Security package separately as part of your application installation.
- **Open Client** — This installation comes with a number of distribution packages that provide client-side security. You must select the appropriate package and include it with your application. For more information about the various distribution packages, see the *Progress Open Client Developer's Guide*.

Once you have installed a client with the Client-Side Security package and have a Secure AppServer, you can take advantage of HTTPS. HTTPS extends HTTP by executing the HTTP protocol across an SSL connection rather than an unencrypted TCP/IP connection. When an SSL connection is established between the AppServer client and the Web server, the HTTP protocol is executed in the context of the encrypted and authenticated channel. Using the HTTPS protocol to connect to a Secure AppServer gives the client the capability of tunneling through firewalls and sending encrypted data.

NOTE: If the client uses HTTP to connect, and, on the server side, the `httpsEnabled` property in the `ubroker.properties` file is set to on (1), and the Secure AppServer package is installed, the AIA instance automatically redirects the client back to the Web server using the same URL except the protocol is changed to HTTPS and the Web server port is changed to the `securePort` property value.

Figure C–2 shows the Secure AppServer Internet Adapter (AIA/S) architecture with HTTPS tunneling running with a Secure AppServer.

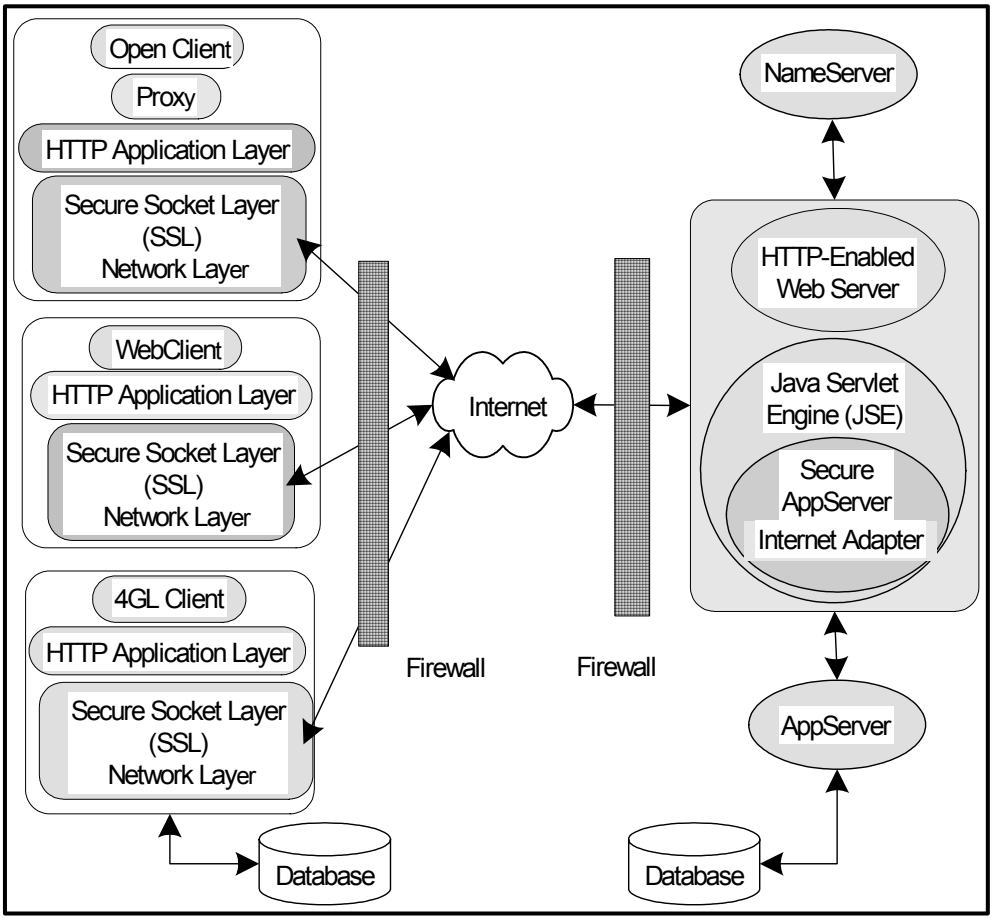


Figure C–2: AppServer Internet Adapter Architecture With HTTPS

When using HTTPS, Progress establishes an SSL connection with the Web server. As part of establishing the connection, Progress verifies the server certificate using the Root CA certificates that have been installed on the client machine. If Progress determines that the server certificate is not valid, the connection to the AIA/S instance is denied.

When checking a certificate, a Progress client:

1. Verifies that the server certificate is signed by one of the trusted Root CA certificates installed on the client machine

2. Verifies that the certificate has not expired by comparing the current time to the timestamps included in the certificate

3. Verifies that the certificate host name in the URL is the same as the common name in the certificate (CN field)

   Step 3 is an optional verification that ensures the host machine that the client connects to is the intended host machine. This verification is done by comparing the host name the user specified in the URL with the host name in the certificate returned by the Web server.

   By default, Progress performs host verification. However, when establishing a connection, a client application might indicate that host verification should not be performed. For WebClients and 4GL clients, a client can indicate that host verification should not be performed by using the `-nohostverify` connection parameter. For Open clients, the client uses the `RunTimeProperties.setNoHostVerify` method on the `runtimeProperties` object to indicate that host verification should not be performed.

If any one of these steps determines that the certification is not valid, then the connection to the AIA/S instance fails.

NOTE: For 4GL and Web Client, the *`Install-Dir`*`/certs` directory contains a readme file that provides instructions for configuring certificates. The Progress installation includes a set of publicly available and well-known CA certificates, such as Verisign©, in the *`Install-Dir`*`/certs` directory. These certificates do not require configuration. If you want to use other certificates, see the readme file. For Open Clients, the `psccerts.zip and Psccertsn.jar` file is also in the *`Install-Dir`*`/certs` directory. For more information about these files, see the *Progress Open Client Developer's Guide*

Supported Client Platforms

For the following clients, HTTPS is supported on these platforms:

- WebClient is supported on the following Windows platforms: 98, NT, ME, and 2000, XP
- 4GL client is supported on all platforms on which Progress currently runs except AIX and all 64 Bit platforms
- Open client is supported on all platforms that have Microsoft or Javasoft compatible JVMs

Supported Server Platforms

On server machines, you can use any Web server that supports HTTPS. For more information, see the "Installing and Configuring Web Servers and Java Servlet Engines" section.

C.1.3 AppServer Internet Adapter and Proxy Servers

An HTTP proxy server is a special type of client-side firewall that limits access to the Internet to the host machine on which the proxy server is running. Other machines that are running within the client network must access the Internet indirectly by going through the proxy server.

Figure C–3 shows the AIA architecture with a proxy server.

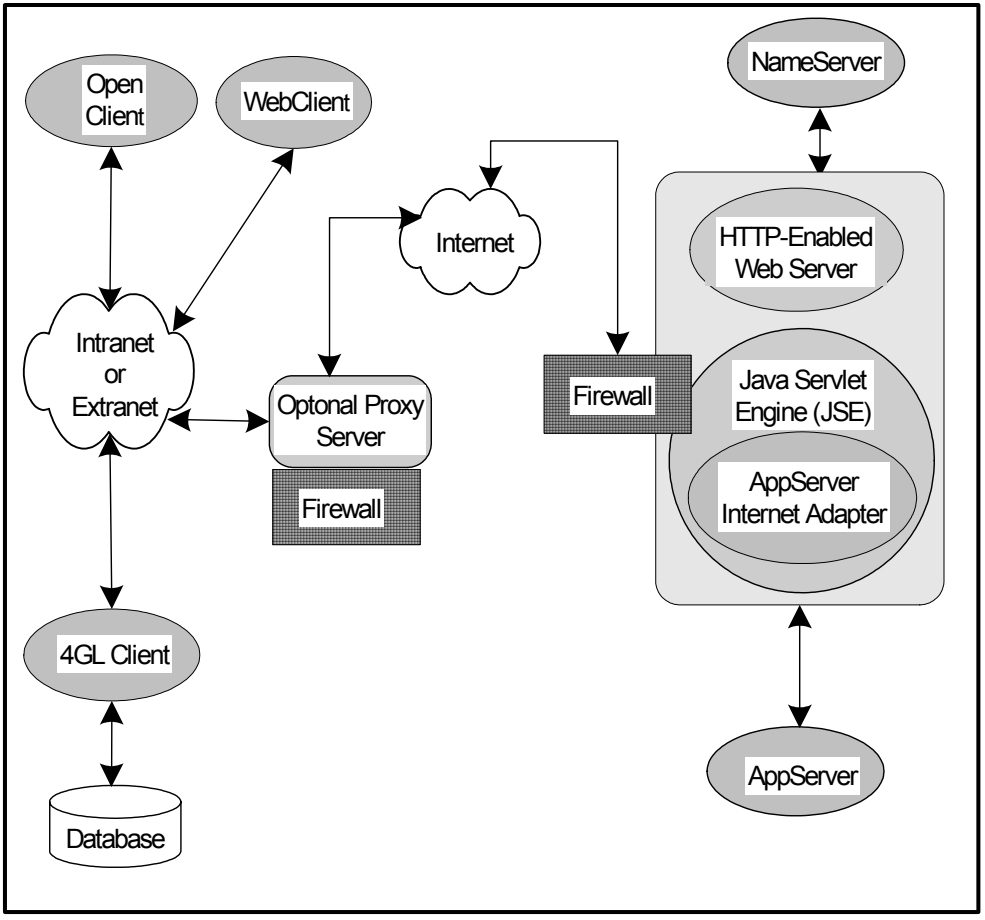


Figure C–3: AIA and HTTP Proxy Server

There are many different types of HTTP proxy servers available. Certain types of HTTP proxy servers require that the client application know the location of the proxy server and establish a connection to the proxy server according to the specification set out in the HTTP standard. If the client network is protected by a proxy server for which an explicit connection is needed, the client application must specify the proxy host, port user, and password.

For WebClients and 4GL clients, the host, port, user, and password are specified using the Proxy Host (`-proxyhost`) and Proxy Port (`-proxyport`) startup parameters. For Open Clients, the host, port, userid, and password are specified using methods on the `runtimeProperties` object.

C.2 Installing and Licensing the AppServer Internet Adapter

You can install AIA when you install the AppServer, or you can install AIA separately on another machine. There are two versions of the AIA product:

- **AIA** — Provides HTTP tunneling to a standard AppServer and allows an unlimited number of user connections.
- **AIA/S** — Provides HTTPS tunneling to a Secure AppServer and monitors the number or user connections.

In addition to installing AIA or AIA/S, the following must be installed on the same machine as the Java servlet engine (JSE) and configured for use with AIA or AIA/S:

- Web server (HTTPS enabled for AIA/S)
- JSE

For information about administering and configuring AIA, see the "Administering and Configuring an AppServer Internet Adapter" section. For information about installing and configuring Web servers, see the "Installing and Configuring Web Servers and Java Servlet Engines" section.

AIA/S uses SSL to support server authentication. Supporting server authentication requires that X.509 digital certificates be installed on both the Web server and the client machine. At each Web server to be accessed, a server certificate that uniquely identifies this Web server must be installed. As part of the SSL protocol, this server certificate is sent from the Web server to the client.

To secure the Web server for deployment, you must perform the following tasks:

- Send a request to a Certificate Authority for a digital certificate.
- Receive the certificate from the Certificate Authority.
- Install the certificate on the Web server.
- Enable encryption on the Web server.

At the client machine, the Root Certification Authority (CA) certificate for all server certificates must be installed before HTTPS can be used. For WebClients and 4GL clients, the Root CA certificates must be installed in the *Install-Dir*/certs directory, where *Install-Dir* identifies the directory where Progress is installed. Root CA certificates for publicly available and well-known CAs, such as Verisign©, are automatically installed into the *Install-Dir*/certs directory during the Progress installation. For more information about digital certificates for Open Client, see the *Progress Open Client Developer's Guide*.

C.3 Installing and Configuring Web Servers and Java Servlet Engines

AIA and AIA/S support the Sun Java Servlet specification Version 2.1 and requires a Web server that is compatible with a Java servlet engine (JSE). The Web server and servlet engine are not provided by Progress. To use AIA, you must install and configure a Web server and a JSE. Table C–1 lists the Web servers, servlet engines, and platforms on which AIA has been qualified. Other JSEs might work, but have not been qualified. If you encounter problems using a JSE other than those listed in Table C–1, you should test with one of the qualified JSEs. For more recent information, see the *Release Notes* and the Progress Knowledge Base on the Web (http://www.Progress.com/support).

Table C–1: Qualified Web Servers and Java Servlet Engines

Web Server	Java Servlet Engine	Platform
Apache	Tomcat	All
iPlanet/Netscape	Jrun™ and ServletExec	All
Internet Information Server (IIS)	Jrun and ServletExec	Windows NT

A JSE must be installed for each Web server that you are using. How the JSE is installed and configured to run AIA or AIA/S depends on the JSE and the Web server. The information you provide during the installation and configuration varies depending on the Web server and JSE. However, the following reflects some of the tasks you must perform and the information you must provide when you install and configure a JSE for the AIA:

- Add *Install-Dir*/java/aia.zip to the JSE CLASSPATH.
- On UNIX platforms only, add *Install-Dir*/lib to the JSE LIBRARYPATH.
- Create a servlet instance for each AIA entry in the ubroker.properties file, and use **com.progress.aia.Aia** for the **ClassName** property. Then use the following values for the **Init Arguments** for each instance of the servlet:
  - *InstallDir* = Directory where Progress is installed
  - *instanceName* = Name that you specified for the AIA instance in the ubroker.properties file.
  - *PropertyFileName* = Location of the ubroker.properties file. This would typically be Progress *Install-Dir*/properties/ubroker.properties.
  - Optionally set up a virtual path extension to point to AIA. For example, you could use a virtual path of /aia/Aia. Progress Software strongly recommends that you define a virtual path.

  NOTE: The values you enter are case sensitive. So you must enter the values to meet the requirements of the platform you use.

- On UNIX only, you must add the following environment variables to the script that starts the JSE:
  - Add:

```
DLC=Install Dir; export DLC
```

  - Add:

```
WRKDIR=Working directory; export WRKDIR
```

NOTE: This is the path to the work directory that you specified when you installed Progress.

  - Add:

```
LD_LIBRARY_PATH= Install-Dir /lib:$LD_LIBRARY_PATH; export
LD_LIBRARY_PATH
```

NOTE: Depending on your operating system, the library path variable might differ (for example, LIBPATH, SHLIB_PATH, and so on).

You must restart your JSE and Web server for these changes to take effect. After you have completed these steps, test the AIA configuration using a Web browser with the following URL:

```
http://host:port/path
```

Where *path* identifies the virtual path and servlet name used to invoke the JSE and the AIA and must include the following information:

- A virtual path that the JSE has configured within the Web server to recognize and pass directly to the JSE instead of looking for that directory on the Web server
- Name of the Java servlet to be invoked by the JSE

The specification of the *path* depends on your Web server and JSE. For more information, see the "Administering and Configuring an AppServer Internet Adapter" section and the documentation for your Web server and servlet engine.

If you used all the recommended settings, the URL is:

```
http://host:port/aia/Aia
```

For instructions on installing and configuring a specific JSE, see the documentation for that JSE. For more specific information about configuring the AIA with JSEs, see the Release Notes and the Progress Knowledge Base on the Web (`http://www.Progress.com/support`). For information about the configuration tasks you must complete to use the AIA with WebClient, see the Progress Knowledge Base on the Web (`http://www.Progress.com/support`).

C.4 Administering and Configuring an AppServer Internet Adapter

You must configure AIA for your specific deployment. A running AIA instance is an extension of the JSE, and the JSE starts and stops the AIA instance when necessary. When the JSE starts an AIA instance on behalf of a client, the AIA instance reads the local `ubroker.properties` file to load its configuration information.

NOTE: The configuration information is loaded the first time that instance of an AIA is started, and the AIA continues to use that configuration information until it is stopped and restarted by the JSE.

C.4.1 ubroker.properties File

The `ubroker.properties` file contains configuration information for each instance of an AIA. The default `ubroker.properties` file contains an AIA root group and an Aia1 instance. All of the properties in the AIA root group are inherited by the Aia1 instance and any other AIA instance that you define. The root AIA group includes the following properties that you can modify:

- logFile
- controllingNameServer
- logAppend
- loggingLevel
- httpsEnabled
- idleConnectionTimeout
- nsClientPortRetryInterval

- nsClientPortRetry
- minNSCClientPort
- maxNSClientPort
- securePort
- soReadTimeout
- AllowAiaCmds
- adminIPList
- registerNameServer
- Host
- Port

NOTE: The `ubroker.properties` file contains a default instance named Aia1.

For more information about these properties and how to set the values for the AIA, see the comments in the `ubroker.properties` file in the `properties` subdirectory of your Progress installation directory, or the Progress Explorer Help.

Editing the ubroker.properties File

On Windows NT, you can use the Progress Explorer to configure AIA properties. The Progress Explorer provides a graphical interface for configuring the values in the `ubroker.properties` file for an AIA instance. On platforms other than Windows NT, you can use any text editor to edit the AIA properties.

NOTE: To use the Progress Explorer on Windows NT, you must have installed the AdminServer at the same site as the AIA. If the AdminServer is not installed, then you must use a text editor to manually edit the `ubroker.properties` file.

Validating Changes To the ubroker.properties File

If you use a tool other than Progress Explorer to edit the values in the `ubroker.properties` file, you can validate the changes made to the `ubroker.properties` file for an AIA instance using the AIACONFIG command. The following shows the syntax for the AIACONFIG command:

SYNTAX

```
aiaconfig [
 [  [ -name AIA-instance-name ]
    [ -propfile path-to-properties-file ]
    [ -validate ]
 ] | -help ]
```

NOTE: If you upgrade to a new version of Progress, you might want to retain changes made to the previous version's `ubroker.properties` file. If that is the case, place the old properties file in the installed properties directory to replace the default. When starting the AdminServer for the first time, a properties file conversion utility automatically runs.

C.5 Viewing AppServer Internet Adapter Connection and Configuration Information

Progress provides utilities to help you test and debug AIA using a browser. You can use your browser and URLs in the following form to view connection and configuration information about an AIA:

SYNTAX

```
http://Host:Port/path/
```

where:

Host identifies the machine where the Web server is running.

Port identifies the port number for the Web server.

Path identifies the virtual path used to invoke the AIA.

NOTE: By default, the `allowAiaCmds` property in the `ubroker.properties` file is turned off (set to 0) for security reasons. If you want to test the AIA using a browser, you must change the `allowAiaCmds` value to 1, and then stop and restart the JSE.

The *path* used to view connection and configuration information depends on how you configure your Web server and Java servlet engine. In these examples, the *path* was configured based on the default configuration used by Progress. The first line verifies that AIA is configured correctly and is being successfully executed by the JSE. If you cannot run this example, there is a problem with the configuration. The second line provides status information about AIA:

```
http://Host:Port/aia/Aia
http://Host:Port/aia/Aia?GetServletStatus
```

C.5.1 Connection Status

The connection status provides dynamic information about each client connection.The following shows sample status information using the following URL:

```
http:/starbuck:84/aia/Aia
```

Figure C–4 shows connection information using the above URL.

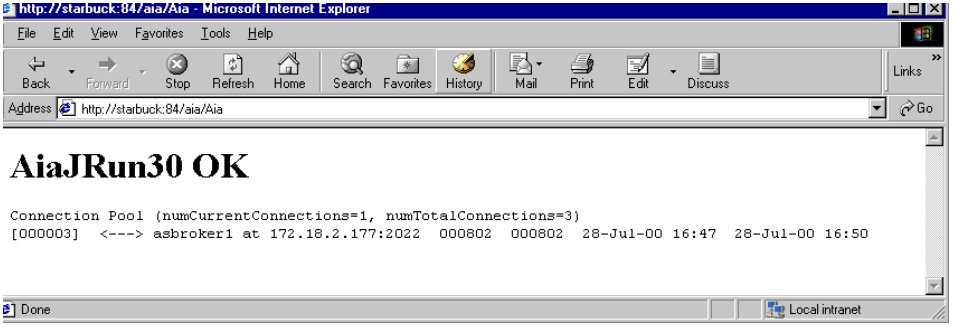


Figure C–4: AIA Client Connection Information

Connection information provides you with information about that AIA configuration. If the AIA is configured correctly, the following information is displayed:

- The first line shows that the AIA servlet was accessed successfully.
- The second line, Connection Pool, shows the current and total connections to the AIA servlet since startup.
- The third line shows the current connection information including the thread number, the AppServer and IP address, total number of connections, and the date and time the servlet was last accessed.

C.5.2 Configuration Information

Configuration information comes from the `ubroker.properties` file. The following shows sample configuration information using the following URL:

```
http://starbuck:84/aia/Aia?GetServletStatus
```

Figure C–5 shows AIA configuration information for the above sample URL.

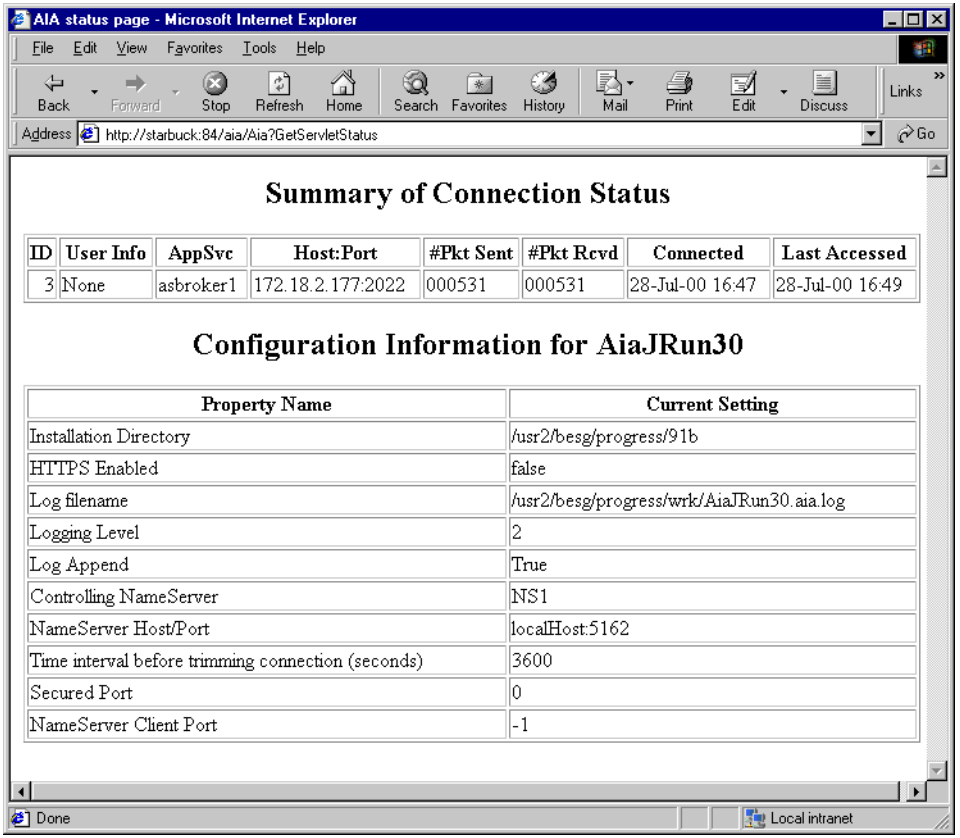


Figure C–5: AIA Configuration Information

The configuration information includes the following:

- **ID** — An identification number used for tracking connections. A connection id is assigned when the connection is made.
- **User Info —** Identifies the user. This information displays either just the user name, the user name and password, or the value None.
- **AppSvc** — Identifies the application service being used by the connection.
- **Host:Port —** host and port of the client process using the connection.
- **#Pkt Sent —** number of packets sent during a connection.
- **#Pkt Rcvd —** number of packets received during a connection.
- **Connected —** time the client connected.
- **Last Accessed** — last time a request was sent or received.

For more information about these properties see the "ubroker.properties File" section and the `ubroker.properties` file in the `properties` directory of your Progress installation.

D

Progress 4GL Language Reference

This appendix documents enhancements and changes made to the Progress 4GL language in Version 9.1D and contains information about the following:

- Browse Widget includes the following new attributes:
  - FIT-LAST-COLUMN Attribute
  - MIN-COLUMN-WIDTH-CHARS Attribute
  - MIN-COLUMN-WIDTH-PIXELS Attribute
  - NO-EMPTY-SPACE Attribute
  - VIEW-FIRST-COLUMN-ON-REOPEN Attribute
- Buffer Object handle includes the following new methods and attributes:
  - AMBIGUOUS Attribute
  - DISABLE-DUMP-TRIGGERS( ) Method
  - DISABLE-LOAD-TRIGGERS( ) Method
  - FIND-CURRENT( ) Method
  - FIND-FIRST Method
  - FIND-LAST Method

  - FIND-UNIQUE Method

- Chained Widget Attributes
- DEFINE BROWSE Statement
- Dynamic Invoke includes the following new 4GL elements:
  - ASYNCHRONOUS Attribute
  - ASYNC-REQUEST-HANDLE Attribute
  - CALL Object Handle
  - CALL-NAME Attribute
  - CALL-TYPE Attribute
  - CREATE CALL Statement
  - CLEAR() Method
  - EVENT-PROCEDURE Attribute
  - EVENT-PROCEDURE-CONTEXT Attribute
  - IN-HANDLE Attribute
  - INVOKE() Method
  - IS-PARAMETER-SET Attribute
  - NUM-PARAMETERS Attribute
  - PERSISTENT Attribute
  - RETURN-VALUE Attribute
  - RETURN-VALUE-DATA-TYPE Attribute
  - SERVER Attribute
  - SET-PARAMETER() Method
- HTML-CHARSET Attribute
- INPUT-VALUE Attribute

- LITERAL-QUESTION Attribute
- Logical Function
- Quoter Function
- SESSION System Handle includes the following new attributes:
  - BASE-ADE Attribute
  - DEBUG-ALERT Attribute
  - PROXY-PASSWORD Attribute
  - PROXY-USERID Attribute
- Simple API for XML (SAX) support for the SAX-ATTRIBUTES Object includes the following new 4GL elements:
  - SAX-ATTRIBUTES Object
  - ADM-DATA Attribute
  - GET-INDEX-BY-QNAME() Method
  - GET-LOCALNAME-BY-INDEX() Method
  - GET-QNAME-BY-INDEX() Method
  - GET-TYPE-BY-INDEX() Method
  - GET-TYPE-BY-NAMESPACE-NAME() Method
  - GET-TYPE-BY-QNAME() Method
  - GET-URI-BY-INDEX() Method
  - GET-VALUE-BY-INDEX() Method
  - GET-VALUE-BY-NAMESPACE-NAME() Method
  - GET-VALUE-BY-QNAME() Method
  - NUM-ITEMS Attribute
  - PROXY-PASSWORD Attribute

- TYPE Attribute
- UNIQUE-ID Attribute

- Simple API for XML (SAX) support for the SAX-READER Object includes the following new 4GL elements:
  - ADM-DATA Attribute
  - CREATE SAX-READER Statement
  - HANDLER Attribute
  - LOCATOR-COLUMN-NUMBER Attribute
  - LOCATOR-LINE-NUMBER Attribute
  - LOCATOR-PUBLIC-ID Attribute
  - LOCATOR-SYSTEM-ID Attribute
  - LOCATOR-LINE-NUMBER Attribute
  - PARSE-STATUS Attribute
  - PROXY-PASSWORD Attribute
  - SAX-PARSE() Method
  - SAX-PARSE-FIRST() Method
  - SAX-PARSE-NEXT() Method
  - SAX-READER Object
  - SERVER Attribute
  - SET-INPUT-SOURCE() Method
  - STOP-PARSING() Method
  - SUPPRESS-NAMESPACE-PROCESSING Attribute
  - UNIQUE-ID Attribute
  - VALIDATION-ENABLED Attribute

- Sockets include the following new methods:
  - ENABLE-CONNECTIONS( ) Method
  - GET-SOCKET-OPTION( ) Method
  - SET-SOCKET-OPTION( ) Method
- XML support for Large Text Nodes and DTD Search paths includes the following new attributes and methods:
  - MEMPTR-TO-NODE-VALUE() Method
  - NODE-VALUE-TO-MEMPTR() Method
  - SCHEMA-PATH Attribute
  - SUPPRESS-NAMESPACE-PROCESSING Attribute
  - XML-SCHEMA-PATH Attribute
  - XML-SUPPRESS-NAMESPACE-PROCESSING Attribute

For more information about language elements, see the *Progress Language Reference*.

ADM-DATA Attribute

An arbitrary string value.

NOTE: The ADM-DATA attribute is for use by the ADM only.

Data Type: CHARACTER

Access: Readable

Applies To: SAX-ATTRIBUTES Object and SAX-READER Object

AMBIGUOUS Attribute

Indicates whether more than one record matched the FIND predicate.

If AMBIGUOUS is TRUE, the most recent unique find on the buffer failed because more than one record matched the FIND predicate. Otherwise, AMBIGUOUS is FALSE.

Return Type: LOGICAL

Applies To: Buffer Object handle

The AMBIGOUS attribute has the following syntax:

SYNTAX

```
AMBIGUOUS
```

The following shows an example of the AMBIGUOUS attribute:

```
DEFINE VARIABLE bh AS HANDLE.
DEFINE VAR myname as char.

bh = BUFFER customer:HANDLE.

bh:FIND-UNIQUE("where name = " + myname).
MESSAGE bh:AMBIGUOUS.
```

SEE ALSO

AMBIGUOUS function in the *Progress Language Reference*.

ASYNCHRONOUS Attribute

Makes a dynamic invoke asynchronous. That is, the INVOKE() method with ASYNCHRONOUS set to TRUE does dynamically what the RUN statement with the ASYNCHRONOUS option does statically.

NOTE: If ASYNCHRONOUS is TRUE, the logic being invoked must reside on a Progress AppServer.

Data Type: LOGICAL

Access: Readable/Writable

Applies To: CALL Object

SYNTAX

```
ASYNCHRONOUS [ = logical-expression ]
```

logical-expression

A LOGICAL expression which, if TRUE, makes the dynamic invoke asynchronous.

The default is FALSE.

If ASYNCHRONOUS is TRUE, when the dynamic invoke returns successfully, it sets the ASYNC-REQUEST-HANDLE attribute to indicate an asynchronous-request object that provides information on this particular asynchronous request.

ASYNC-REQUEST-HANDLE Attribute

A handle to an asynchronous-request object providing information on an asynchronous invoke.

NOTE: Applies only if ASYNCHRONOUS is TRUE.

Data Type: HANDLE

Access: Read Only

Applies To: CALL Object

The default is ? (the unknown value).

BASE-ADE Attribute

A new SESSION attribute (SESSION:BASE-ADE) and a new startup parameter (`-baseADE`), specifies where the client should look for the ADE r-code.

With the BASE-ADE session attribute and the `-baseADE` parameter, you can have multiple versions of ADE r-code on the same machine, and you can easily switch between them.

For more information about the `-baseADE` startup parameter, see *Progress Startup Command and Parameter Reference*.

The BASE-ADE attribute is used with the SESSION System Handle to set the location of the ADE r-code directory. When set, Progress adds the directory, followed by all the procedure libraries in the directory to the PROPATH.

Data Type: Character

Access: Readable/Writeble

Applies To: SESSION System Handle

NOTES

- If `-baseADE` is not specified at startup, SESSION:BASE-ADE is UNKNOWN (?) until it is set.
- When SESSION:BASE-ADE is set, Progress adds the directory, followed by all of the procedure libraries in the directory to PROPATH. If the directory or any of the procedure libraries are already on PROPATH, Progress does not add them.

- When SESSION:BASE-ADE is set, Progress removes all PROPATH entries representing the current ADE r-code directory and procedure libraries before adding the new PROPATH entries. Progress adds the new PROPATH entries at the location where it removed the previous entries. Progress only removes PROPATH entries that it added. For example, if `$DLC/gui/adecomm.pl` is part of the PROPATH, it remains on the PROPATH after BASE-ADE is set to a directory other than `$DLC/gui`.

- If BASE-ADE is set to an empty string, Progress removes whatever it added to PROPATH.

- Progress does not remove the ADE r-code directory or any of the procedure libraries in that directory from PROPATH, even if the PROPATH statement does not contain them. These entries are part of the base PROPATH. If `-baseADE` or SESSION:BASE-ADE is used, the directory and procedure libraries that Progress adds are part of the base PROPATH and remain part of the PROPATH even if the PROPATH statement does not contain them.

- SESSION:BASE-ADE modifies PROPATH. If the old PROPATH contains a procedure library that is not in the new PROPATH, Progress automatically closes the procedure library as long as there are no procedures from the library running.

CALL-NAME Attribute

The name of one of the following:

- An external procedure, internal procedure, or user-defined function you want to invoke dynamically
- An attribute you want to get or set dynamically
- A method you want to run dynamically

SYNTAX

```
CALL-NAME [ = char-expression ]
```

char-expression

A CHARACTER expression indicating the name of the procedure, user-defined function, or attribute.

Data Type: CHARACTER

Access: Readable/Writable

Applies To: CALL Object

The default is ? (the unknown value).

If *char-expression* is a procedure, the rules for finding the procedure are the same as those of the RUN statement, including the treatment of SUPER. However, the actual instance of the file, procedure name, or attribute is not determined until the dynamic invoke is executed.

If *char-expression* is a user-defined function, an attribute, or a method, the rules for finding the function or attribute are the same as those for finding a function or attribute invoked statically.

CALL Object Handle

Interfaces	OS	SpeedScript
All	All	Yes

A handle to a CALL object, which lets you do the following dynamically:

- Invoke an external procedure, internal procedure, or user-defined function
- Get or set an attribute
- Run a method

Invoking logic dynamically requires many more lines of code than does invoking it statically, which can make your program hard to read. For this reason, Progress Software Corporation recommends that you invoke logic dynamically only when absolutely necessary. Specifically, use the CALL object only when you cannot use the RUN statement, the DYNAMIC-FUNCTION() function, or *widget*:*attribute* or *widget*:*method* syntax, as in the following situations:

- To invoke an internal or external procedure whose *calling sequence* (number of parameters and the data type of each) is unknown at compile time

  NOTE: But if only the **name** of the procedure is unknown at compile time, use the RUN statement with the VALUE option — and avoid the CALL object all together.

- To invoke a function whose calling sequence is unknown at compile time

  NOTE: But if only the **name** of the function is unknown at compile time, use the DYNAMIC-FUNCTION() function — and avoid the CALL object altogether.

- To reference a widget attribute or method whose name is unknown at compile time

  If you already know the name of the attribute or procedure, you know its syntax, since the name implies certain syntax. And if you know the syntax, you know the calling sequence, since the syntax defines the calling sequence. And if you know the calling sequence, you can use *widget:attribute* or *widget:method* syntax — and avoid the CALL object altogether.

The CALL object handle has attributes and methods that tell CALL what to do, how to do it, and when to do it.

SYNTAX

```
call-object-handle [ :attribute | :method ]
```

call-object-handle

A handle to a CALL object.

attribute

An attribute of the CALL object. Table D–1 summarizes each attribute.

Table D–1: CALL Object Attributes *(1 of 4)*

Attribute	Type	Description
ASYNCHRONOUS Attribute	LOGICAL	(Applies only when the logic to be dynamically invoked resides on a Progress AppServer.) Indicates whether you want the logic to run asynchronously.
ASYNC-REQUEST-HANDLE Attribute	HANDLE	(Applies only when ASYNCHRONOUS is TRUE.) A handle to an asynchronous-request object containing detailed information on your asynchronous request. Set by INVOKE() when ASYNCHRONOUS is TRUE.

Table D–1: CALL Object Attributes *(2 of 4)*

Attribute	Type	Description
CALL-NAME Attribute	CHARACTER	The name of one of the following: • A procedure or user-defined function you want to invoke dynamically • An attribute or you want to get or set dynamically • A method you want to invoke dynamically
CALL-TYPE Attribute	INTEGER	Whether you are dynamically: • Invoking a procedure • Invoking a user-defined function • Getting an attribute or invoking a method • Setting an attribute
EVENT-PROCEDURE Attribute	CHARACTER	(Applies only when ASYNCHRONOUS is TRUE.) The name of an internal procedure you want executed when a dynamic, asynchronous invoke returns. **NOTE:** You must set EVENT-PROCEDURE before you run INVOKE().

Table D–1: CALL Object Attributes *(3 of 4)*

Attribute	Type	Description
EVENT-PROCEDURE-CONTEXT Attribute	HANDLE	(Applies only when ASYNCHRONOUS is TRUE.) A handle to a running persistent procedure containing an internal procedure you want executed when a dynamic, asynchronous invoke returns.
IN-HANDLE Attribute	HANDLE	Contains one of the following: • A handle to a persistent procedure you just invoked dynamically • A handle to an already-running persistent procedure containing logic you want to invoke dynamically • A handle to an object whose attributes or methods you want to invoke dynamically
IS-PARAMETER-SET Attribute	LOGICAL	Whether you have already set a particular parameter.
NUM-PARAMETERS Attribute	INTEGER	The number of parameters you are passing.
PERSISTENT Attribute	LOGICAL	Whether you want an external procedure you plan to invoke dynamically to run persistently.

Table D–1: CALL Object Attributes *(4 of 4)*

Attribute	Type	Description
RETURN-VALUE Attribute	RETURN-VALUE-DATA-TYPE	(Applies only to user-defined functions, attributes, and methods) One of the following: • The value returned by a user-defined function • The value of an attribute • The value returned by a method
RETURN-VALUE-DATA-TYPE Attribute	CHARACTER	(Applies only to user-defined functions, attributes, and methods) The data type of the return value. If you set this before you execute INVOKE(), the return value is converted to this data type. If you do not set this before you execute INVOKE(), it indicates the return value's data type.
SERVER Attribute	HANDLE	A handle to an AppServer containing logic you want to invoke dynamically.

method

A method of the CALL object. The methods let you set parameters, reset attributes to their default values, and invoke the CALL object.

Table D–2 briefly describes each method.

Table D–2: CALL Object Methods

Method	Return Type	Description
CLEAR() Method	LOGICAL	Resets each attribute of the CALL object to its default value. **NOTE:** To reset just the parameters, set the NUM-PARAMETERS attribute to zero.
INVOKE() Method	None (similar to the RUN statement)	Lets you dynamically do the following: • Invoke an external procedure, internal procedure, or user-defined function • Get or set an attribute • Invoke a method **NOTE:** INVOKE() cannot occur within an expression.
SET-PARAMETER() Method	LOGICAL	Lets you set parameters of one of the following: • A procedure or user-defined function you want to invoke dynamically • An attribute you want to get or set dynamically • A method you want to invoke dynamically

EXAMPLES

The following fragment dynamically invokes an external procedure nonpersistently:

```
DEFINE VARIABLE hCall AS HANDLE.

CREATE CALL hCall.

/*invoke hello.p nonpersistently */
hCall:CALL-NAME = "hello.p".
hCall:NUM-PARAMETERS = 1.
hCall:SET-PARAMETER( 1, "CHARACTER", "INPUT", "HELLO WORLD").
hCall:INVOKE.

/* clean up */
DELETE OBJECT hCall.
```

The following fragment dynamically invokes an external procedure persistently, then dynamically invokes one of its internal procedures:

```
DEFINE VARIABLE hCall AS HANDLE.

CREATE CALL hCall.

/*invoke persis.p persistently */
hCall:CALL-NAME = "persis.p".
hCall:PERSISTENT = true.
HCall:INVOKE.
   /* IN-HANDLE is automatically set
     to the handle of the persistent procedure
  */

/* invoke internal-persis-proc in persis.p */
HCall:CALL-NAME = "internal-persis-proc".
hCall:NUM-PARAMETERS = 1.
hCall:SET-PARAMETER( 1, "INTEGER", "INPUT", 333).
hCall:INVOKE.

/* clean up */
DELETE PROCEDURE hCall:IN-HANDLE.
DELETE OBJECT hCall.
```

The following fragment uses a single CALL object handle multiple times:

```
DEFINE VARIABLE hCall AS HANDLE.

CREATE CALL hCall.

/*invoke hello.p nonpersistently */
hCall:CALL-NAME = "hello.p".
hCall:NUM-PARAMETERS = 1.
hCall:SET-PARAMETER( 1, "CHARACTER", "INPUT", "HELLO WORLD").
hCall:INVOKE.

/* reset the CALL object handle */
hCall:CLEAR.

/*invoke persis.p persistently */
hCall:CALL-NAME = "persis.p".
hCall:PERSISTENT = true.
HCall:INVOKE.
   /* IN-HANDLE is automatically set
      to the handle of the persistent procedure
  */

/* invoke internal-persis-proc in persis.p
HCall:CALL-NAME = "internal-persis-proc".
hCall:NUM-PARAMETERS = 1.
hCall:SET-PARAMETER( 1, "INTEGER", "INPUT", 333).
hCall:INVOKE.

/* clean up */
DELETE PROCEDURE hCall:IN-HANDLE.
DELETE OBJECT hCall.
```

The following fragment gets an attribute:

```
/* get title of frame */

hCall:IN-HANDLE = myframe_handle.
HCALL:CALL-TYPE = GET-ATTR-CALL-TYPE.
hCall:CALL-NAME = "TITLE".
hCall:INVOKE.
Mytitle = hCall:RETURN-VALUE.
```

The following fragment sets an attribute:

```
/* set SESSION:NUMERIC-FORMAT to "european" */

hCall:IN-HANDLE = "session".
HCALL:CALL-TYPE = SET-ATTR-CALL-TYPE.
hCall:CALL-NAME = "numeric-format".
hCall:NUM-PARAMETERS = 1.
hCall:SET-PARAMETER( 1, "CHAR", "INPUT", "european").
hCall:INVOKE.
```

The following fragment drives the INVOKE() from a TEMP-TABLE:

```
/*suppose hRuntt is a temp-table that has one record with the
following fields:
   parm_1
   parm_2
   ...
   parm_n
   run-name
   nparms
   datatypes, extent nparms
   iomodes, extent nparms
*/

DEFINE INPUT PARAMETER TABLE-HANDLE hRuntt.

DEFINE VARIABLE hDtypes  AS HANDLE.
DEFINE VARIABLE hIOmodes AS HANDLE.
DEFINE VARIABLE hCall    AS HANDLE.
DEFINE VARIABLE i        AS INTEGER.

hDtypes  = hRuntt:BUFFER-FIELD("datatypes").
hIOmodes = hRuntt:BUFFER-FIELD("iOmodes").

hRuntt:FIND-FIRST.

CREATE CALL hCall.

hCall:CALL-NAME = hRuntt:BUFFER-FIELD("run-name"):BUFFER-VALUE.

hCall:NUM-PARAMETERS = hRuntt:BUFFER-FIELD("nparms"):BUFFER-VALUE.

FOR i = 1 to hCall:NUM-PARAMETERS.
    hCall:SET-PARAMETER( i,
           hDtypes:BUFFER-VALUE(i),
           hIOmodes:BUFFER-VALUE(i),
           hRuntt:BUFFER-FIELD(i):BUFFER-VALUE).
END.

hCall:INVOKE.
DELETE OBJECT hCall.
/* if there are output parms,
    get the values from hRuntt:BUFFER-FIELD(i)
*/
```

NOTES

- To create a CALL object, use the following syntax:

  SYNTAX

  ```
  CREATE object handle [ IN widget-pool ]
  ```

  For example:

  ```
  CREATE CALL hCall.
  ```

  NOTE: Unlike most 4GL objects, the CALL object, by default, goes into the SESSION widget pool, not into the closest unnamed widget pool.

- To delete a CALL object, use the following syntax:

  SYNTAX

  ```
  DELETE OBJECT handle
  ```

  For example:

  ```
  DELETE OBJECT CALL hCall.
  ```

  Since the CALL object, by default, goes into the SESSION widget pool, not into the closest unnamed widget pool, to delete a CALL object created when the IN widget-pool option is not used, use the `DELETE object handle` syntax explicitly,

- Invoking logic dynamically requires more lines of code than does invoking logic statically and can make your program hard to read. For this reason, Progress Software Corporation recommends that you invoke logic dynamically only when invoking it statically is impossible.

SEE ALSO

RUN Statement in the *Progress Language Reference*.

CALL-TYPE Attribute

The type of call, which must be the dynamic version of one of the following:

- Invoking an external or internal procedure
- Invoking a user-defined function
- Getting an attribute
- Setting an attribute
- Invoking a method

Data Type: INTEGER

Access: Readable/Writable

Applies To: CALL Object

You can use a keyword constant or an integer. Table D–3 lists the keyword constants and integers you can use.

Table D–3: Keyword Constants For the CALL-TYPE Attribute

Keyword Constant	Integer
PROCEDURE-CALL-TYPE	1
FUNCTION-CALL-TYPE	2
GET-ATTR-CALL-TYPE	3
SET-ATTR-CALL-TYPE	4

The default is PROCEDURE-CALL-TYPE.

To get the value of an attribute or to invoke a method, set CALL-TYPE to GET-ATTR-CALL-TYPE.

To set the value of *object:attribute* to the first parameter set by the SET-PARAMETER() method, set CALL-TYPE to SET-ATTR-CALL-TYPE, as in the following fragment:

```
/* set SESSION:NUMERIC-FORMAT to "european" */

hCall:IN-HANDLE = "session".
HCALL:CALL-TYPE = SET-ATTR-CALL-TYPE.
hCall:CALL-NAME = "numeric-format".
hCall:NUM-PARAMETERS = 1.
hCall:SET-PARAMETER( 1, "CHAR", "INPUT", "european").
hCall:INVOKE.
```

Chained Widget Attributes

Supplies multiple attributes for a widget. When you use this syntax, the middle attributes and methods must have a data type of HANDLE.

The Chained Widget attribute has the following syntax:

SYNTAX

```
{widget-name-reference | handle}: <attribute-name> [:attribute-name]...
```

`widget-name-reference`

Specifies a widget object. For example, SELF, CURRENT-WINDOW, BUFFER customer:HANDLE, or Query q:handle.

`handle`

Variable of type HANDLE.

`attribute-name`

Any attribute for the widget or handle immediately preceding it.

EXAMPLE

```
DEFINE VARIABLE hBuff as HANDLE.
CREATE BUFFER hBuff  FOR TABLE "customer".
MESSAGE hBuff:BUFFER-FIELD(3):NAME.
```

SEE ALSO

WIDGET PHRASE, Widget function, HANDLES in the *Progress Language Reference*.

CLEAR() Method

Resets each attribute of a CALL object to its default.

This lets a CALL object be reused.

Return Type: LOGICAL

Applies To: CALL Object Handle

SYNTAX

```
CLEAR()
```

NOTE: To clear just the parameters, set the NUM-PARAMETERS attribute to zero.

CREATE CALL Statement

Interfaces	OS	SpeedScript
All	All	Yes

Creates a CALL object, then stores a handle to it in the handle variable specified.

The CALL object, its attributes, and its methods, are used by applications to invoke logic dynamically.

SYNTAX

```
CREATE CALL handle [ IN widget-pool ] [ NO-ERROR ]
```

handle

A HANDLE expression that indicates the name of a HANDLE variable into which a handle to the new CALL object is stored.

widget-pool

A CHARACTER expression that indicates the name of the widget pool to contain the new CALL object.

NO-ERROR

Suppresses reporting of errors that occur while CREATE CALL executes. Afterwards, you can get information on possible errors by checking the ERROR-STATUS system handle.

NOTES

- Unlike most 4GL objects, the CALL object, by default, is assigned not to the closest unnamed widget pool, but rather to the SESSION widget pool.
- A CALL object is deleted automatically when its widget pool is deleted. To delete it earlier than its widget pool, use the DELETE OBJECT statement, specifying the handle to the CALL object, as in the following fragment:

```
DELETE OBJECT hMyCallObj.
```

SEE ALSO

CALL Object Handle in this bulletin and the DELETE OBJECT Statement in the *Progress Language Reference*.

CREATE SAX-READER Statement

Interfaces	OS	SpeedScript
All	All	Yes

Creates an instance of a SAX-READER object and assigns its handle to the handle variable you specify. The SAX-READER object can be used to control the parsing of XML source.

SYNTAX

```
CREATE SAX-READER handle
  [ IN WIDGET-POOL pool-name ] [ NO-ERROR ]
```

handle

A variable of type HANDLE into which CREATE SAX-READER stores the new handle.

pool-name

A CHARACTER expression indicating the widget pool in which the object is created. If you do not specify a widget pool, the object is created in the current default widget pool.

NO-ERROR

Causes any errors that occur during the creation of the SAX-READER handle to be suppressed. After the CREATE SAX-READER statement completes, check the ERROR-STATUS handle for information concerning any errors that might have occurred.

DEBUG-ALERT Attribute

Use the DEBUG-ALERT attribute on the SESSION system handle along with the new `-clientlog` startup parameter to access 4GL stack trace information during a session.

Data Type: LOGICAL

Access: Readable/Writeable

Applies To: SESSION System Handle

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

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

You can also set Debug Alert to TRUE using the `-debugalert` startup parameter. For more information about the `-clientlog` and the `-debugalert` startup parameter, see the *Progress Startup Command and Parameter Reference*.

DEFINE BROWSE Statement

The DEFINE BROWSE statement includes new options that let you control how the browse displays. The options for the DEFINE BROWSE Statement are also attributes for the Browse widget. For more information, see:

- FIT-LAST-COLUMN Attribute
- NO-EMPTY-SPACE Attribute

DISABLE-DUMP-TRIGGERS( ) Method

The DISABLE-DUMP-TRIGGERS method, which works like the DISABLE TRIGGERS FOR DUMP statement, allows a user to access a buffer object's table without firing FIND triggers.

Return Type: LOGICAL

Applies To: Buffer Object handle

DISABLE-DUMP-TRIGGERS has the following syntax:

SYNTAX

```
DISABLE-DUMP-TRIGGERS ()
```

NOTES

- To run this method, the end user must have CAN-DUMP privileges for the table.
- You cannot disable triggers from a persistent procedure.
- The FIND trigger remains disabled until the procedure in which it is disabled returns.

SEE ALSO

The DUMP option for the DISABLE TRIGGERS statement in the *Progress Language Reference*.

DISABLE-LOAD-TRIGGERS( ) Method

The DISABLE-LOAD-TRIGGERS method, which works like the DISABLE TRIGGERS FOR LOAD statement, allows a user to subsequently create or update a buffer object's table without firing update triggers such as CREATE, WRITE, ASSIGN, or DELETE.

Return Type: LOGICAL

Applies To: Buffer Object handle

DISABLE-LOAD-TRIGGERS has the following syntax:

SYNTAX

```
DISABLE-LOAD-TRIGGERS (allow-replication)
```

`allow-replication`

A logical expression that allows the following replication triggers to run when you set *`allow-replication`* to true: REPLICATION-CREATE, REPLICATION-DELETE, and REPLICATION-WRITE.

NOTES

- To run this method, the end user must have CAN-LOAD privileges for the table.
- You cannot load triggers from a persistent procedure.
- The triggers remain disabled until the procedure in which it is disabled returns.

SEE ALSO

The LOAD option for the DISABLE TRIGGERS statement in the *Progress Language Reference*.

ENABLE-CONNECTIONS( ) Method

Specifies the TCP/IP port that Progress should use to listen for new connections. Once called, Progress will automatically listen for and accept new connections for the specified port.

Return Type: LOGICAL

Applies To: Server-socket Object Handle

SYNTAX

```
ENABLE-CONNECTIONS( connection-parms )
```

connection-parms

A character string expression that contains a space-separated list of one or more socket connection parameters.

Table D–4 describes each socket connection parameter, which can be included in this string.

Table D–4: Socket Connection Parameters *(1 of 2)*

Parameter	Description
-S *socket-port*	The TCP/IP port number that Progress should listen to and accept connections on. You can specify either an explicit port number or a TCP service name. If you use a TCP service name, the method uses the port number associated with that name in the TCP/IP services file.

Table D–4: Socket Connection Parameters *(2 of 2)*

Parameter	Description
-pf *filename*	Optional. A text file containing any of the socket connection parameters described in this table. If this file contains any other Progress startup parameters, this method ignores them.
-qsize *backlog*	Optional. The length of the pending-connection queue — that is, the maximum number of connection requests you want the server to queue while it processes the connections already accepted. If the queue is full when a connection request is received, it is refused. The default length of the queue depends on the platform. **NOTE:** On some platforms, the value you supply for *backlog* is modified by addition, subtraction, multiplication, division, or some combination of these, and it is this modified value that becomes the maximum length of the queue. For more information, see the documentation for your platform.

Neither an AppServer nor a WebSpeed agent can act as a socket server, since they are already listening on a port. ENABLE-CONNECTIONS is only valid from batch clients, GUI clients and character clients. This method will generate an error if it is called from an invalid application. This method will also generate an error if it is called multiple times without the DISABLE-CONNECTION( ) method being called in between.

EVENT-PROCEDURE Attribute

The name of an internal procedure you want Progress to execute when a dynamic, asynchronous invoke returns.

NOTE: Applies only if the ASYNCHRONOUS attribute is TRUE.

SYNTAX

```
EVENT-PROCEDURE [ = event-internal-procedure ]
```

event-internal-procedure

A CHARACTER expression indicating the name of the event handler for a dynamic invoke that is asynchronous.

The default is ? (the unknown value).

NOTE: If the ASYNCHRONOUS attribute is TRUE, you must set EVENT-PROCEDURE before you execute the INVOKE() method.

EVENT-PROCEDURE-CONTEXT Attribute

A handle to a running persistent procedure containing an internal procedure you want Progress to execute when a dynamic, asynchronous invoke returns.

NOTE: Applies only if the ASYNCHRONOUS attribute is TRUE.

Data Type: HANDLE

Access: Readable/Writable

Applies To: CALL Object

SYNTAX

```
EVENT-PROCEDURE-CONTEXT [ = handle-expression ]
```

handle-expression

A HANDLE expression indicating the handle of a running persistent procedure in whose context an event procedure is to run.

The default value is ? (the unknown value).

FIND-CURRENT( ) Method

Changes the *lockmode* of a record in a buffer.

Return Type: LOGICAL

Applies To: Buffer Object handle

The FIND-CURRENT method has the following syntax:

SYNTAX

```
FIND-CURRENT ( [ lockmode [ , wait-mode ] ] )
```

lockmode

An integer expression evaluating to one of the following constants: SHARE-LOCK, EXCLUSIVE-LOCK, or NO-LOCK. You can assign any of these constants to an integer variable. For example, mylock = NO-LOCK.

The default is SHARE-LOCK.

waitmode

An integer expression evaluating to one of the following: NO-WAIT, 0, or ?. You can assign NO-WAIT to an integer variable. For example, mywait = NO-WAIT.

The default is to wait.

The following shows an example of the FIND-CURRENT method:

```
DEFINE VARIABLE bh AS HANDLE.

bh = BUFFER customer:HANDLE.

DO Transaction:
 bh:FIND-CURRENT(EXCLUSIVE-LOCK).
END.
 MESSAGE bh:CURRENT-CHANGED cust.cust-num.
```

NOTES

- If the change in lock status succeeds, TRUE is returned, otherwise FALSE is returned.
- If the lock change fails, a message displays. You can suppress the message using NO-ERROR on the statement containing the method.
- Executing the FIND-CURRENT method resets the CURRENT-CHANGED attribute. If the record in the database changes between the time the original record was found and the time the FIND-CURRENT executes, the CURRENT-CHANGED attribute returns TRUE. If the record does not change, then the CURRENT-CHANGED attribute returns FALSE.

SEE ALSO

FIND-BY-ROWID method, FIND CURRENT statement, CURRENT-CHANGED attribute, CURRENT-CHANGED function, in the *Progress Language Reference*.

FIND-FIRST Method

Gets a single record. This method lets a user get the first record that satisfies the predicate expression.

Return Type: LOGICAL

Applies To: Buffer Object handle

The FIND-FIRST method has the following syntax:

SYNTAX

```
FIND-FIRST (predicate-expression [ , lockmode [ , wait-mode ] ] )
```

predicate-expression

A character expression that evaluates to the following syntax:

```
[ WHERE [  logical-expression   ] ]  [ USE-INDEX index-name  ]
```

Once evaluated, *predicate-expression* can contain only constants and unabbreviated references to fields from the buffer.

The *predicate-expression* itself can be built using a concatenation of character expressions.

lockmode

An integer expression evaluating to one of the following constants: SHARE-LOCK, EXCLUSIVE-LOCK, or NO-LOCK. You can assign any of these constants to an integer variable. For example, mylock = NO-LOCK.

The default is SHARE-LOCK.

wait-mode

An integer expression evaluating to one of the following: NO-WAIT, 0, or ?. You can assign NO-WAIT to an integer variable. For example, mywait = NO-WAIT.

The default is to wait.

The following shows some examples of FIND-FIRST method:

```
DEFINE VARIABLE bh AS HANDLE.
DEFINE VARIABLE myname AS char.

bh = BUFFER customer:HANDLE.

bh:FIND-FIRST("where cust-num > 2", NO-LOCK).
do transaction ;
  bh:FIND-FIRST("", EXCLUSIVE-LOCK).
end.
bh:FIND-FIRST("where name = " + QUOTER(myname) , NO-LOCK).
```

NOTES

- If FIND-FIRST succeeds, it returns TRUE, otherwise it returns FALSE.
- If FIND-FIRST fails, it does not raise an error but displays a message. You can suppress this message by using NO-ERROR on the statement containing the method.

SEE ALSO

FIND-UNIQUE Method, FIND-LAST Method, FIND-CURRENT( ) Method, FIND LAST statement, FIND FIRST statement, and FIND-BY-ROWID method in the *Progress Language Reference*.

FIND-LAST Method

Gets a single record. This method lets a user get the last record that satisfies the predicate expression.

Return Type: LOGICAL

Applies To: Buffer Object handle

The FIND-LAST method has the following syntax:

SYNTAX

```
FIND-LAST  (predicate-expression [ , lockmode [ , wait-mode ] ] )
```

predicate-expression

A character expression that evaluates to the following syntax:

```
[ WHERE [  logical-expression   ] ]  [ USE-INDEX index-name  ]
```

Once evaluated, *predicate-expression* can contain only constants and unabbreviated references to fields from the buffer.

The *predicate-expression* itself can be built using a concatenation of character expressions.

lockmode

An integer expression evaluating to one of the following constants: SHARE-LOCK, EXCLUSIVE-LOCK, or NO-LOCK. You can assign any of these constants to an integer variable. For example, mylock = NO-LOCK.

The default is SHARE-LOCK.

wait-mode

An integer expression evaluating to one of the following: NO-WAIT, 0, or ?. You can assign NO-WAIT to an integer variable. For example, mywait = NO-WAIT.

The default is to wait.

The following shows some examples of the FIND-LAST method:

```
DEFINE VARIABLE bh AS HANDLE.

bh = BUFFER customer:HANDLE.

bh:FIND-LAST("where balance > 0 use-index name").
bh:FIND-LAST("where cust-num > 5 and address < 'z'").
```

NOTES

- If FIND-LAST succeeds, it returns TRUE, otherwise it returns FALSE.
- If FIND-LAST fails, it does not raise an error but displays an error message. You can suppress the message using NO-ERROR on the statement containing the method.

SEE ALSO

FIND-UNIQUE Method, FIND-FIRST Method, FIND-CURRENT( ) Method, and FIND LAST statement in the *Progress Language Reference*.

FIND-UNIQUE Method

Gets a single record. This method lets a user get a unique record that satisfies the predicate expression

Return Type: LOGICAL

Applies To: Buffer Object handle

The FIND-UNIQUE method has the following syntax:

SYNTAX

```
FIND-UNIQUE (predicate-expression [ , lockmode [ , wait-mode ] ] )
```

predicate-expression

A character expression that evaluates to the following syntax:

```
[ WHERE [ logical-expression ] ] [ USE-INDEX index-name ]
```

Once evaluated, *predicate-expression* can contain only constants and unabbreviated references to fields from the buffer.

The *predicate-expression* itself can be built using a concatenation of character expressions.

lockmode

An integer expression evaluating to one of the following constants: SHARE-LOCK, EXCLUSIVE-LOCK, or NO-LOCK. You can assign NO-LOCK to an integer variable. For example, mylock = NO-LOCK.

The default is SHARE-LOCK.

waitmode

An integer expression evaluating to one of the following: NO-WAIT, 0, or ?. The default is to wait.

You can assign NO-WAIT to an integer variable. For example, mywait = NO-WAIT.

The following shows some examples of the FIND-UNIQUE method:

```
DEFINE VARIABLE bh AS HANDLE.
DEFINE VAR myname as char

bh = BUFFER customer:HANDLE.


bh:FIND-UNIQUE("where cust-num < 3 and name = 'lift line skiing'").

bh:FIND-UNIQUE("where cust-num = 30").

bh:FIND-UNIQUE("where name = " + QUOTER(myname).
MESSAGE bh:AMBIGUOUS.
```

NOTES

- If FIND-UNIQUE succeeds, it returns TRUE, otherwise it returns FALSE.
- If FIND-UNIQUE fails, a message displays. You can suppress this message using NO-ERROR on the statement containing the method.
- If more than one record satisfies the predicate expression, then the AMBIGUOUS attribute is set to TRUE.

SEE ALSO

FIND-FIRST Method, FIND-LAST Method, FIND-CURRENT( ) Method in this bulletin, and the FIND Statement in the *Progress Language Reference*

FIT-LAST-COLUMN Attribute (Graphical interfaces only)

The new FIT-LAST-COLUMN option for the DEFINE BROWSE statement and corresponding Browse widget attribute, allows the browse to be displayed so that there is no empty space to the right and no horizontal scroll bar by potentially widening or shrinking the last browse column's width.

When this option is specified, and the last browse column can be fully or partially displayed in the browse's viewport, then the last browse column's width is adjusted so that it fits within the viewport with no empty space to its right and no horizontal scroll bar.

If the last browse column is fully contained in the viewport with empty space to its right, it grows so that its right edge is adjacent to the vertical scroll bar.

If the last browse column extends outside the viewport, it shrinks so its right edge is adjacent to the vertical scroll bar and the horizontal scroll bar is not needed.

The default value for the FIT-LAST-COLUMN attribute is FALSE.

Data Type: LOGICAL

Access: Readable/Writeble

Applies To: Browse Widget

The FIT-LAST-COLUMN attribute performs the same function as the EXPANDABLE attribute. Progress software 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.

The following shows the DEFINE BROWSE statement syntax with FIT-LAST-COLUMN specified:

SYNTAX

```
DEFINE BROWSE b1 QUERY q1
DISPLAY customer.cust-num customer.name
ENABLE customer.cust-num WITH  3 DOWN WIDTH 40 FIT-LAST-COLUMN
```

NOTES

- The MIN-COLUMN-WIDTH attribute affects the FIT-LAST-COLUMN attribute. As a result, if FIT-LAST-COLUMN is set to TRUE, the last browse column is resized to fit within the viewport only if its width is no smaller than the minimum width. To specify the minimum size that the last browse column's width can be reduced to, use the MIN-COLUMN-WIDTH-PIXELS or MIN-COLUMN-WIDTH-CHARS attribute. See MIN-COLUMN-WIDTH-PIXELS Attribute and MIN-COLUMN-WIDTH-CHARS Attribute for more information.

- FIT-LAST-COLUMN and NO-EMPTY-SPACE are mutually exclusive. If both are specified in the DEFINE BROWSE statement, the compiler displays an error message. If one attribute is set to TRUE while the other attribute is already TRUE, a warning message displays at run time.

- FIT-LAST-COLUMN is primarily intended for use in the initial layout of a static browse. It is most useful when laying out a browse with a specified width when you have only a few browse columns and you want to fully use the available space in your viewport.

- If the FIT-LAST-COLUMN attribute is set to TRUE, and, subsequently, any browse column's width is changed or the browse's width is changed, then the last browse column's width might be adjusted so that it fits within the viewport with no empty space and no horizontal scroll bar.

- When the last browse column's width is set at run time after the browse is realized, then FIT-LAST-COLUMN is ignored.

- If the FIT-LAST-COLUMN attribute is set to FALSE, the last browse column's width remains the same and is never changed by Progress.

- The new FIT-LAST-COLUMN attribute and the existing EXPANDABLE attribute have the same behavior. 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 Software 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.

GET-INDEX-BY-NAMESPACE-NAME() Method

Returns the 1-based index of the attribute with the given namespace name.

Return Type: INTEGER

Applies To: SAX-ATTRIBUTES Object

SYNTAX

```
GET-INDEX-BY-NAMESPACE-NAME ( uri, localname )
```

uri

The namespace URI (that is, the URI to which the attribute's prefix refers), or, if the name has no namespace URI, an empty string.

localname

The local (unqualified) name of the attribute.

Returns ? (the unknown value) if no attribute's uri-localname combination matches the combination of *uri* and *localname*.

GET-INDEX-BY-QNAME() Method

Returns the 1-based index of the attribute that has the given XML qualified name.

Return Type: INTEGER

Applies To: SAX-ATTRIBUTES Object

SYNTAX

```
GET-INDEX-BY-QNAME ( qname )
```

qname

A CHARACTER expression indicating the XML qualified name of the attribute of interest.

Returns ? (the unknown value) if no attribute's XML qualified name matches *qname*.

GET-LOCALNAME-BY-INDEX() Method

Returns the local (unqualified) name of the attribute at the given 1-based index.

Return Type: CHARACTER

Applies To: SAX-ATTRIBUTES Object

SYNTAX

```
GET-LOCALNAME-BY-INDEX ( index )
```

index

An INTEGER expression indicating the 1-based index of the attribute.

Looks up an attribute's local (unqualified) name by index. Returns ? (the unknown value) if namespace processing is disabled, if the index is less than 1, or if the index is greater than the value of SAX-ATTRIBUTES's NUM-ITEMS attribute.

GET-QNAME-BY-INDEX() Method

Returns the XML qualified name of the attribute at the given 1-based index.

Return Type: CHARACTER

Applies To: SAX-ATTRIBUTES Object

SYNTAX

```
GET-QNAME-BY-INDEX ( index )
```

index

An INTEGER expression indicating the 1-based index of the attribute.

Looks up an attribute's XML qualified name by index. Returns ? (the unknown value) if the index is less than 1 or greater than the value of SAX-ATTRIBUTES's NUM-ITEMS attribute.

GET-SOCKET-OPTION( ) Method

If TRUE, returns a comma separated string containing values appropriate for the socket option specified; otherwise, returns the unknown value (?).

Return Type: CHARACTER

Applies To: Socket Object Handle

SYNTAX

```
GET-SOCKET-OPTION( name )
```

name

A character expression indicating the name of the socket option to be retrieved. Table D–5 describes the options Progress supports.

Table D–5: Options For GET-SOCKET-OPTION() *(1 of 2)*

Option	Value Returned
TCP-NODELAY	TRUE if the *enable indicator* is on; FALSE otherwise.
SO-LINGER	Two values: the *onoff indicator* followed by the *linger time*.
SO-KEEPALIVE	TRUE if the option is on; FALSE otherwise. The default depends on how the socket object was created: • For socket objects created using CREATE SOCKET, the default is off. • For socket objects created by Progress and passed as a parameter to the event-procedure context, the default is on.
SO-RESUEADDR	TRUE if the option is on; FALSE otherwise. The default depends on the platform.

Table D–5: Options For GET-SOCKET-OPTION() *(2 of 2)*

Option	Value Returned
SO-RCVBUF	An integer that indicates the size of the receive buffer. The default depends on the platform.
SO-SNDBUF	An integer that indicates the size of the send buffer. The default depends on the platform.
SO-RCVTIMEO	The timeout length — that is, the number of seconds you want the socket to wait for expected data before timing out. The default is -1, which tells the socket to wait forever. **NOTE:** The timeout length is not guaranteed to be precise to the second.

This method returns option-specific data if the retrieval of the option succeeded and the unknown (?) value otherwise. An error can occur if:

- *name* is not a Progress-supported socket option.
- Getting the socket option fails.

GET-TYPE-BY-INDEX() Method

Returns the type of the attribute at the given 1-based index.

Return Type: CHARACTER

Applies To: SAX-ATTRIBUTES Object

SYNTAX

```
GET-TYPE-BY-INDEX ( index )
```

index

An INTEGER expression indicating the 1-based index of the attribute.

The attribute type is one of the following strings: “CDATA,” “ID,” “IDREF,” “IDREFS,” “NMTOKEN,” “NMTOKENS,” “ENTITY,” “ENTITIES,” or “NOTATION.” These are always uppercase.

Returns ? (the unknown value) if the index is less than 1 or greater than the value of SAX-ATTRIBUTES’s NUM-ITEMS attribute.

GET-TYPE-BY-NAMESPACE-NAME() Method

Returns the type of the attribute that has the given namespace name.

Return Type: CHARACTER

Applies To: SAX-ATTRIBUTES Object

SYNTAX

```
GET-TYPE-BY-NAMESPACE-NAME ( uri, localname )
```

uri

The namespace URI (that is, the URI to which the attribute’s prefix refers), or, if the name has no namespace URI, an empty string.

localname

The local (unqualified) name of the attribute.

The attribute type is one of the following strings: “CDATA,” “ID,” “IDREF,” “IDREFS,” “NMTOKEN,” “NMTOKENS,” “ENTITY,” “ENTITIES,” or “NOTATION.” These are always uppercase.

Returns ? (the unknown value) if no attribute’s XML qualified name matches the combination of *uri* and *localname* or if namespace processing is disabled.

GET-TYPE-BY-QNAME() Method

Returns the type of the attribute that has the given XML qualified name.

Return Type: CHARACTER

Applies To: SAX-ATTRIBUTES Object

SYNTAX

```
GET-TYPE-BY-QNAME ( qname )
```

qname

A CHARACTER expression indicating the XML qualified name of the attribute of interest.

The attribute type is one of the following strings: “CDATA,” “ID,” “IDREF,” “IDREFS,” “NMTOKEN,” “NMTOKENS,” “ENTITY,” “ENTITIES,” or “NOTATION.” These are always uppercase.

Returns ? (the unknown value) if no attribute’s XML qualified name matches *qname*.

GET-URI-BY-INDEX() Method

Returns the namespace URI of the attribute at the given 1-based index.

Return Type: CHARACTER

Applies To: SAX-ATTRIBUTES Object

SYNTAX

```
GET-URI-BY-INDEX ( index )
```

index

An INTEGER expression indicating the 1-based index of the attribute.

Returns ? (the unknown value) if *index* is less than 1 or greater than the value of SAX-ATTRIBUTES's NUM-ITEMS attribute.

GET-VALUE-BY-INDEX() Method

Returns the value of the attribute at the given 1-based index.

Return Type: CHARACTER

Applies To: SAX-ATTRIBUTES Object

SYNTAX

```
GET-VALUE-BY-INDEX ( index )
```

index

An INTEGER expression indicating the 1-based index of the attribute.

Returns ? (the unknown value) if *index* is less than 1 or greater than the value of SAX-ATTRIBUTES's NUM-ITEMS attribute.

GET-VALUE-BY-NAMESPACE-NAME() Method

Returns the value of the attribute with the given namespace name.

Return Type: CHARACTER

Applies To: SAX-ATTRIBUTES Object

SYNTAX

```
GET-VALUE-BY-NAMESPACE-NAME ( uri, localname )
```

uri

The namespace URI (that is, the URI to which the attribute's prefix refers), or, if the name has no namespace URI, an empty string.

localname

The local (unqualified) name of the attribute.

Returns ? (the unknown value) if no attribute has a namespace name that matches the *uri* and *localname*, or if namespace processing is disabled.

GET-VALUE-BY-QNAME() Method

Returns the value of the attribute with the given XML qualified name.

Return Type: CHARACTER

Applies To: SAX-ATTRIBUTES Object

SYNTAX

```
GET-VALUE-BY-QNAME ( qname )
```

qname

A CHARACTER expression indicating the XML qualified name of the attribute.

Returns ? (the unknown value) if no attribute has an XML qualified name that matches *qname*.

HANDLER Attribute

A handle to the procedure containing the SAX callbacks.

Data Type: HANDLE

Access: Readable/Writable

Applies To: SAX-READER Object

The default is a handle to the procedure that contains the SAX-PARSE(), or the SAX-PARSE-FIRST() and the SAX-PARSE-NEXT().

When SAX-PARSE(), SAX-PARSE-FIRST(), or SAX-PARSE-NEXT() executes, the SAX-READER object looks for callbacks only in the procedure whose handle is stored in HANDLER. HANDLER must be a valid procedure handle and cannot be a proxy.

NOTE: It is permissible for both the driver procedure (the procedure containing the SAX-PARSE(), or the SAX-PARSE-FIRST() and SAX-PARSE-NEXT()) and handler procedure of a SAX application to reside on a remote AppServer. If this occurs, callbacks are invoked local to the AppServer.

Callbacks can reside within a special handler procedure file that is run persistently or within the driver procedure, which is the procedure that contains the SAX-PARSE(), or the SAX-PARSE-FIRST() and SAX-PARSE-NEXT().

Within a procedure file, to get a handle to the procedure file, use the THIS-PROCEDURE handle. The following fragment assigns HANDLER a handle to the current procedure:

```
hSaxReader:HANDLER = THIS-PROCEDURE.
```

HTML-CHARSET Attribute (WebSpeed only)

The Progress version (as opposed to the MIME version) of the code-page name of a Web request. Set by Progress for each Web request when a WebSpeed application incorporates dynamic code-page support.

Data Type: CHARACTER

Access: Readable/Writable

Applies To: WEB-CONTEXT Handle

The default is “”, the empty string.

When Progress sets HTML-CHARSET, each time a Web browser sends a Web request to the application:

1. When the request is received by the WebSpeed Agent, Progress converts it from the HTML-CHARSET code page to the Agent’s CPINTERNAL code page.

2. When the Agent responds to the request with a Web page, Progress converts the Web page from the Agent’s CPINTERNAL code page to the HTML-CHARSET code page.

CAUTION: If the application modifies the contents of HTML-CHARSET, dynamic code-page support might fail.

For more information on dynamic code-page support, see the *WebSpeed Installation and Configuration Guide* and the *WebSpeed Product Update Bulletin*.

IN-HANDLE Attribute

A handle to one of the following:

- A persistent procedure you just started up dynamically (by invoking an external procedure with the PERSISTENT attribute set to TRUE)
- A running persistent procedure containing an internal procedure or user-defined function you want to invoke dynamically
- An object whose attributes you want to get or set dynamically, or whose methods you want to run dynamically

Data Type: HANDLE

Access: Readable/Writable

Applies To: CALL Object

SYNTAX

```
IN-HANDLE [ = { handle-expression | char-expression } ]
```

handle-expression

A HANDLE expression. Table D–6 explains what the handle can indicate and who sets it.

Table D–6: What IN-HANDLE Indicates and Who Sets It

If the Handle Indicates...	It Is Set By...
A persistent procedure you just started up dynamically	The INVOKE() method
An already-running persistent procedure containing logic (in the form of internal procedures and user-defined functions) you want to invoke dynamically	You
An object whose attributes you want to get or set dynamically	You

char-expression

A CHARACTER expression indicating the name of a system object, such as “SESSION” or “FILE-INFO.”

The default is ? (the unknown value).

When you use IN-HANDLE to call an internal procedure, IN-HANDLE affects INVOKE() the same way the IN *proc-handle* phrase affects the RUN statement. Similarly, when you use IN-HANDLE to call a user-defined function, it affects INVOKE() the same way the IN *proc-handle* phrase affects the DYNAMIC-FUNCTION function. In both cases, IN-HANDLE specifies the instance of the external procedure that contains the internal procedure or user-defined function.

When IN-HANDLE is used to get or set an attribute or to invoke a method, it represents a handle to the object the attribute applies to. If the attribute applies to a system object such as the SESSION handle or the FILE-INFO handle, IN-HANDLE can be set to a character string such as “SESSION” or “FILE-INFO” that indicates the name of the system object.

NOTE: When you create a running persistent procedure by running an external procedure persistently, you can do this statically or dynamically. Similarly, you can run any of the persistent procedure’s internal procedures and user-defined functions statically or dynamically.

For information on dynamically invoking logic that resides on a Progress AppServer, see the SERVER Attribute reference entry in this bulletin.

INPUT-VALUE Attribute

Used for data-representation widgets, such as field-level widgets that represent variables or database fields. The value for the INPUT-VALUE attribute is the unformatted SCREEN-VALUE of a widget.

For any widget that has a SCREEN-VALUE, INPUT-VALUE returns the unformatted value of the widget's SCREEN-VALUE, in the native data type of the widget.

Data Type: Same as the field or variable associated with the widget

Access: Readable

Applies To: Browse-cell, Combo-box, Editor, Fill-in, Literal, Radio-set, Selection-list, Slider, Text, Toggle-box

The INPUT-VALUE attribute has the following syntax:

SYNTAX

```
<widget-name>:INPUT-VALUE
```

NOTES

- When you use the INPUT-VALUE attribute, there are no formatting characters and the data type is the data type of the associated field. The relationship between a field's INPUT-VALUE and the value in the record buffer is the same as the relationship between the SCREEN-VALUE and the record buffer. For example, changing the record buffer does not affect the INPUT-VALUE.
- When you use the DISPLAY, PUT, or MESSAGE statements for the INPUT-VALUE of a widget that has the DATE data type, you see the formatted value because these statements automatically format DATE values. To display an unformatted date, first assign INPUT-VALUE to an INT variable, and then display the variable.

SEE ALSO

SCREEN-VALUE Attribute in the *Progress Language Reference*.

INVOKE() Method

Lets you do the following dynamically:

- Invoke an external procedure, internal procedure, or user-defined function
- Get or set an attribute
- Run a method

Return Type: None (Similar to the RUN statement)

Applies To: CALL Object Handle

SYNTAX

```
INVOKE()
```

To determine what action to take, INVOKE() examines the CALL-NAME and CALL-TYPE attributes.

Before you execute INVOKE(), you must set the CALL-NAME attribute.

Table D–7 describes what INVOKE() does for each call type.

Table D–7: What INVOKE() Does For Each Call Type *(1 of 2)*

For This Call Type...	INVOKE()...
PROCEDURE-CALL-TYPE (the default)	Follows the rules of the RUN statement to determine what to invoke and whether to invoke external or internal procedures.
FUNCTION-CALL-TYPE	Follows the rules of the DYNAMIC-FUNCTION() function.

Table D–7: What INVOKE() Does For Each Call Type *(2 of 2)*

For This Call Type...	INVOKE()...
GET-ATTR-CALL-TYPE	Follows the rules of *widget:attribute*. **NOTE:** The IN-HANDLE attribute must be set before INVOKE() is executed.
SET-ATTR-CALL-TYPE	Sets the attribute specified by the CALL-NAME attribute to the first parameter specified by the SET-PARAMETER() method. **NOTE:** The IN-HANDLE attribute must be set before INVOKE() is executed.

If the PERSISTENT attribute is TRUE (which is valid only for invoking an external procedure), the procedure runs persistently, and when INVOKE() returns, IN-HANDLE contains a handle to the persistent procedure.

Before you dynamically invoke an external procedure that is remote — that is, one that resides on an AppServer — you must set the SERVER attribute to the handle of the AppServer.

When INVOKE() starts executing, it examines the NUM-PARAMETERS attribute. If NUM-PARAMETERS is nonzero, INVOKE() uses each parameter set by the SET-PARAMETER() method, even one set during a previous use of the CALL object.

NOTE: To clear all parameters, even those set during a previous use of the CALL object, set the NUM-PARAMETERS attribute to zero.

No parameters are evaluated during INVOKE() processing. Parameters are evaluated only during SET-PARAMETER() processing.

When the invoked routine starts, if any parameter indicated by the NUM-PARAMETERS attribute has not been set, you will get an error message.

For more information on INVOKE(), see the reference entries for the RUN Statement and the DYNAMIC-FUNCTION() Function.

NOTE: INVOKE() cannot occur within an expression.

IS-PARAMETER-SET Attribute

Whether you have already set a parameter.

Data Type: LOGICAL

Access: Readable

Applies To: CALL Object

SYNTAX

```
IS-PARAMETER-SET(parameter-number)
```

parameter-number

An INTEGER expression indicating the order of the parameter, where 1 represents the first parameter, 2 represents the second parameter, etc.

If the parameter is set, IS-PARAMETER-SET returns TRUE. Else, it returns FALSE.

The default is FALSE.

LITERAL-QUESTION Attribute

Lets a user specify how the “?” is interpreted during an assignment into a BUFFER-VALUE of a BUFFER-FIELD object.

This attribute allows the user to change the default behavior for assigning the “?” value into the BUFFER-VALUE attribute of a BUFFER-FIELD object, if the data type of the field represented by the BUFFER-FIELD object is CHARACTER.

Return Type: LOGICAL

Access: Readable/Writeble

Applies To: Buffer-Field Object handle

When assignments are made into a BUFFER-VALUE attribute of a character BUFFER-FIELD object, enclosing quotes and insertion characters such as $ are removed if the field is CHARACTER. As a result, "?" and ? are treated the same. In some cases users might not want the quotes ("") removed when they want the literal question mark instead of the UNKNOWN (?) value. Therefore, use the LITERAL-QUESTION attribute to indicate whether the question mark is treated as a literal character rather than the UNKNOWN value.

The LITERAL-QUESTION attribute has the following syntax:

SYNTAX

```
LITERAL-QUESTION
```

If this attribute is:

- TRUE, and the data type of the field is CHARACTER, then the literal string "?" becomes the value of the field instead of being set to the UNKNOWN value.
- FALSE, the default value, then the field is set to UNKNOWN whenever a "?" is assigned into it.

LOCATOR-COLUMN-NUMBER Attribute

The current column in the XML source.

Data Type: INTEGER

Access: Readable

Applies To: SAX-READER Object

Valid only in a callback. Use the SELF handle to get it, as in the following fragment:

```
myColNum = SELF:LOCATOR-COLUMN-NUMBER.
```

Gives the column number where the text that caused the current callback ends. The first column in a line is 1.

If the current location is an *external entity* — that is, external to the main XML source — the column number is relative to the beginning of the line in the external entity.

LOCATOR-LINE-NUMBER Attribute

The current line in the XML source.

Data Type: INTEGER

Access: Readable

Applies To: SAX-READER Object

Valid only in a callback. Use the SELF handle to get it, as in the following fragment:

```
myLineNum = SELF:LOCATOR-LINE-NUMBER.
```

Gives the line number where the text that caused the current callback ends. The first line in a document is 1.

If the current location is an *external entity* — that is, external to the main XML source — the line number is relative to the beginning of the external entity.

LOCATOR-PUBLIC-ID Attribute

The public identifier of the current XML source.

Data Type: CHARACTER

Access: Readable

Applies To: SAX-READER Object

Valid only in a callback. Use the SELF handle to get it, as in the following fragment:

```
myPublicID = SELF:LOCATOR-PUBLIC-ID.
```

Gives the public identifier of the XML source.

LOCATOR-PUBLIC-ID may refer to an external DTD or to another external entity.

LOCATOR-SYSTEM-ID Attribute

The system identifier of the current XML source.

Data Type: CHARACTER

Access: Readable

Applies To: SAX-READER Object

Valid only in a callback. Use the SELF handle to get it, as in the following fragment:

```
mySystemID = SELF:LOCATOR-SYSTEM-ID.
```

Gives the system identifier of the XML source.

LOCATOR-SYSTEM-ID may refer to an external DTD or to another external entity.

Logical Function

Interface	OS	Speedscript
All	All	Yes

Converts any data type into the Logical data type.

The Logical function has the following syntax:

SYNTAX

```
LOGICAL (<expression> [, <char-expression-format>]  )
```

expression

An *expression* in the datatype that you want to convert to logical.

char-expression-format

A character expression that evaluates to a valid logical format, such as "si/no", or "da/nyet". This argument is ignored unless <*expression*> is of CHARACTER type. Also, this argument is only needed if <*expression*> evaluates to something other than the usual TRUE or FALSE, or YES or NO values.

EXAMPLE

```
DEFINE VARIABLE mychar AS CHAR.
DEFINE VARIABLE v-log AS LOGICAL.
 mychar="si".
v-log=LOGICAL (mychar, "si/no")
/*v-log is TRUE*/
```

NOTES

- The Logical function returns UNKNOWN if the <*expression*> is UNKNOWN.
- If <*expression*> is of type DECIMAL, INTEGER, DATE, or HANDLE, the function returns TRUE if the value of <*expression*> is nonzero. If the value of <*expression*> is 0, it returns FALSE. The second argument is ignored if present.
- If <*expression*> is of type CHARACTER, it returns TRUE or FALSE depending on the value in the expression and the format used. Whether or not <*char-expression-format*> is given, the case-insensitive values TRUE, FALSE, YES, NO, abbreviated to 1 character, are always accepted. For example, a "Y" is interpreted as TRUE.

  If <*char-expression-format*> is given, it is validated. If it is not valid, an error message appears and UNKNOWN is returned. Otherwise, the format is used to interpret the character string if it is not one of the following: TRUE, FALSE, YES, or NO. For example, LOGICAL ("si", "si/no") returns TRUE.
- On Data types such as RAW, MEMPTR, LVARBINARY, and so on, the Logical function returns UNKNOWN, but this is not considered an error.

SEE ALSO

STRING, INTEGER, DATE in the *Progress Language Reference*.

MEMPTR-TO-NODE-VALUE() Method

Sets the value of an X-NODEREF node to the contents of a MEMPTR.

Return Type: LOGICAL

Applies To: X-NODEREF Object Handle

SYNTAX

```
MEMPTR-TO-NODE-VALUE( memptr )
```

memptr

An expression of type MEMPTR.

MEMPTR-TO-NODE-VALUE() raises an error if any of the following occurs:

- The node is read only.
- The node contains invalid XML characters.
- The data in *memptr* is not null-terminated.
- *memptr* is not set to the exact size of the valid data.

For more information on accessing XML documents using the Document Object Model (DOM) interface, see the *Progress External Program Interfaces* book.

MIN-COLUMN-WIDTH-CHARS Attribute

Sets the minimum width of a browse column in character units. If the browse:

- Has **not** been realized, all browse columns will be minimally this size when realized.
- Has been realized, any browse column smaller than the specified minimum is increased to the minimum width.

The default value is equivalent to 1 pixel and depends on the display resolution and the size of the default font of the system.

Data Type: DECIMAL

Access: Readable/Writable

Applies To: Browse widget

NOTES

- An error occurs if you attempt to programmatically set the width of a browse column to a value smaller than the value specified with the MIN-COLUMN-WIDTH-CHARS attribute.
- If COLUMN-RESIZABLE is set to TRUE, the user cannot change a column's width to be less than the minimum width specified with the MIN-COLUMN-WIDTH-CHARS attribute.
- The MIN-COLUMN-WIDTH CHARS attribute affects the FIT-LAST-COLUMN attribute. Therefore, if you set FIT-LAST-COLUMN to TRUE, the last browse column is only resized to fit within the viewport if its width is no smaller than the minimum width. See the "FIT-LAST-COLUMN Attribute" section for more information about the FIT-LAST-COLUMN attribute.
- When you assign a decimal value to an attribute representing a measurement in character units, Progress automatically rounds the assigned value to the nearest decimal value that corresponds to whole pixel units.

MIN-COLUMN-WIDTH-PIXELS Attribute

Sets the minimum width of a browse column in pixels. If the browse:

- Has **not** been realized, all browse columns will be minimally this size when realized.
- Has been realized, any browse column smaller than the specified minimum is increased to the minimum width.

The default minimum browse column width is 1 pixel.

Data Type: INTEGER

Access: Readable/Writable

Applies To: Browse

NOTES

- An error occurs if you attempt to programmatically set the width of a browse column to a value smaller than the value specified using the MIN-COLUMN-WIDTH-PIXELS attribute.
- If COLUMN-RESIZABLE is set to TRUE, the user cannot change a column's width to be less than the minimum width specified with the MIN-COLUMN-WIDTH-PIXELS attribute.
- The MIN-COLUMN-WIDTH PIXELS attribute affects the FIT-LAST-COLUMN attribute. As a result, if FIT-LAST-COLUMN is set to TRUE, the last browse column is only resized to fit within the viewport if its width is no smaller than the minimum width. See the FIT-LAST-COLUMN Attribute section for more information about the FIT-LAST-COLUMN.

NODE-VALUE-TO-MEMPTR() Method

Copies the contents of an XML X-NODEREF node to a MEMPTR. This makes it easier to manipulate when its length exceeds the Progress text string limit, which is approximately 32K.

Return Type: LOGICAL

Applies To: X-NODEREF Object Handle

SYNTAX

```
NODE-VALUE-TO-MEMPTR( memptr )
```

memptr

An expression of type MEMPTR.

NODE-VALUE-TO-MEMPTR() frees the memory currently allocated by *memptr* (if any), allocates sufficient memory to the MEMPTR to accommodate the node, and copies the node to the MEMPTR.

If X-NODEREF:NODE-VALUE is "" (the empty string), the resulting *memptr* has a size of zero.

NOTE: When you no longer need the memory used by *memptr*, you must free it yourself. To do so, use the SET-SIZE statement.

The following fragment uses NODE-VALUE-TO-MEMPTR() to access a large text node:

```
IF my-xnoderef:SUBTYPE = "#TEXT" THEN DO:
  IF my-xnoderef:LENGTH > 32000 THEN DO:
    success = my-xnoderef:NODE-VALUE-TO-MEMPTR(my-memptr).
    pos = 1.
    len = 32000.
    DO WHILE pos < GET-SIZE(my-memptr):
      my-nodeval = GET-STRING(my-memptr, pos, len).
      /* do something with my-nodeval */
      pos = pos + len.
    END.
    SET-SIZE(my-memptr) = 0.
  END.
  ELSE
    my-nodeval = my-xnoderef:NODE-VALUE.
END.
```

For more information on accessing XML documents using the Document Object Model (DOM) interface, see the *Progress External Program Interfaces* book.

NO-EMPTY-SPACE Attribute

The NO-EMPTY-SPACE option for the DEFINE BROWSE statement and the corresponding Browse widget attribute, allows the browse to display with no empty space to the right and no horizontal scroll bar.

When the last browse column can be fully displayed in the browse viewport with empty space to the right, you can use NO-EMPTY-SPACE to widen the width of the last browse column so that the column fits within the viewport with no empty space to its right and no horizontal scroll bar.

The default value for the NO-EMPTY-SPACE is FALSE.

Data Type: LOGICAL

Access: Readable/Writeble

Applies To: Browse widget

The following shows the DEFINE BROWSE statement syntax with NO-EMPTY-SPACE specified:

SYNTAX

```
DEFINE BROWSE b1 QUERY q1
DISPLAY customer.cust-num customer.name
ENABLE customer.cust-num WITH  3 DOWN WIDTH 40 NO-EMPTY-SPACE
```

NOTES

- You cannot specify both NO-EMPTY-SPACE and FIT-LAST-COLUMN for the DEFINE BROWSE statement. If you specify both, the compiler displays an error message. If you set one of these attributes to TRUE and the other attribute is already TRUE, a warning message displays at run time.
- NO-EMPTY-SPACE is primarily intended for use in the initial layout of a static browse. It is most useful when laying out a browse with a specified width when you have only a few browse columns, and you want to fully use the available space in your viewport.
- If the NO-EMPTY-SPACE is set to TRUE and there is empty space, the last browse column is widened to fill up the space. Also, if any browse column's width attribute is changed or the browse's width attribute is changed so that the last browse column is fully displayed in the browse's viewport with empty space to its right, then the last browse column's width is widened so that it fits within the viewport with no empty space and no horizontal scroll bar.

- NO-EMPTY-SPACE never reduces the width of the last browse column.
- Under the following circumstances, NO-EMPTY-SPACE is ignored:
  - When the last browse column's width is explicitly set at run time after the browse is realized.
  - When the last browse column displays partially or entirely outside of the viewport.
- If NO-EMPTY-SPACE is set to FALSE, the last browse column's width remains the same and is never changed by Progress.
- If you specify NO-EMPTY-SPACE for an individual browse, and the `-expandbrow` startup parameter is also specified, then the NO-EMPTY-SPACE attribute overrides `-expandbrow` for that browse.
- When you use NO-EMPTY-SPACE, the original width of the last browse column is not remembered. For example, if the original width of the last browse column is 48 pixels, the DEFINE BROWSE statement has NO-EMPTY-SPACE specified, and if at run time there are 12 blank pixels as empty space in the right side of the viewport, the last browse column's width is increased to 60 pixels, so there is no blank space in the browse.

  If later at run time, the width of a column other than the last column is increased, a horizontal scroll bar is added to the browse. The width for the last browse column remains at 60 pixels.

NUM-ITEMS Attribute

The number of XML attributes contained in a SAX-ATTRIBUTES object.

Data Type: INTEGER

Access: Readable

Applies To: SAX-ATTRIBUTES Object

NUM-PARAMETERS Attribute

The number of parameters expected.

Data Type: INTEGER

Access: Readable/Writable

Applies To: CALL Object

SYNTAX

```
NUM-PARAMETERS(integer-expression)
```

integer-expression

An INTEGER expression indicating the number of parameters expected.

The default is zero.

Whenever NUM-PARAMETERS is set, all existing parameters, including those from earlier uses of SET-PARAMETER, are cleared and deallocated.

If there are parameters to be passed, NUM-PARAMETERS must be set before the INVOKE() method is executed.

When you are getting a parameter, set NUM-PARAMETERS to the actual number of parameters to be passed, which might vary if one or more trailing parameters are optional.

When you are setting an attribute, use NUM-PARAMETERS 1.

PARSE-STATUS Attribute

The current state of a SAX parse.

Data Type: INTEGER

Access: Readable

Applies To: SAX-READER Object

The default value is SAX-UNINITIALIZED.

The values that PARSE-STATUS can assume are described in Table D–8.

Table D–8: Values Of the SAX-READER's PARSE-STATUS Attribute

This Value...	Indicates...
SAX-UNINITIALIZED	No parsing has occurred.
SAX-RUNNING	Parsing has begun.
SAX-COMPLETE	Parsing has begun and one of the following has occurred: • The parser has determined that there are no more tokens in the XML source. • The application has stopped the parser by calling the SAX-READER's STOP-PARSING() method.
SAX-PARSER-ERROR	One of the following has occurred: • The parser could not start or could not continue. Perhaps the parser could not be loaded, the XML source could not be found, the XML source was invalid, etc. • The parser started, but a callback executed a RETURN ERROR statement.

PERSISTENT Attribute

Whether you want an external procedure you are invoking dynamically to run persistently.

Data Type: LOGICAL

Access: Readable/Writable

Applies To: CALL Object

SYNTAX

```
PERSISTENT [ = logical-expression ]
```

logical-expression

A LOGICAL expression indicating whether you want an external procedure invoked dynamically to run persistently.

The default is FALSE.

If PERSISTENT is TRUE, when the dynamic invoke returns, the IN-HANDLE attribute contains a handle to the running persistent procedure.

PRIVATE-DATA Attribute

An arbitrary string associated with a SAX-ATTRIBUTES object.

Data Type: CHARACTER

Access: Readable/Writable

Applies To: SAX-ATTRIBUTES Object and SAX-READER Object

PROXY-PASSWORD Attribute

Authenticates the AppServer client to the HTTP-based proxy server.

Data Type: CHARACTER

Access: Readable/Writeble

Applies To: SESSION System Handle

NOTES

- This attribute corresponds to the `-proxyPassword` startup parameter. If `-proxyPassword` is not specified on the command line, this attribute is UNKNOWN (?) until it is set.
- PROXY-PASSWORD is validated during the AppServer CONNECT() method. If PROXY-PASSWORD is invalid, the CONNECT() method fails and Progress issues an error message.
- If PROXY-USERID **is not** UNKNOWN and PROXY-PASSWORD **is** UNKNOWN, the AppServer CONNECT() method uses a blank proxy password.
- If SESSION:PROXY-USERID is unknown, the CONNECT() method ignores any value of PROXY-PASSWORD.
- PROXY-PASSWORD must be a string of up to 512 printable ASCII characters.

SEE ALSO

- PROXY-USERID Attribute.
- Proxy Password (`-proxyPassword`) in the *Progress Startup Command and Parameter Reference*.
- Proxy Host (`-proxyhost`) in the *Progress Startup Command and Parameter Reference*.
- Proxy Port (`-proxyport`) in the *Progress Startup Command and Parameter Reference*.
- The "Proxy Authentication For WebClient" section in Appendix D, "Progress 4GL Language Reference" of this bulletin.
- *Progress on the Web*

PROXY-USERID Attribute

Authenticates an AppServer client to the HTTP-based proxy server.

Data Type: CHARACTER

Access: Readable/Writeble

Applies To: SESSION System Handle

NOTES

- This attribute corresponds to the `-proxyUserid` startup parameter. If `-proxyUserid` is not specified on the command line, this attribute is UNKNOWN (?) until it is set.
- PROXY-USERID is validated during the AppServer CONNECT() method. If PROXY-USERID is invalid, the CONNECT() method fails and Progress issues an error message.
- PROXY-USERID must be a string of up to 512 printable ASCII characters, including the space character.

SEE ALSO

- PROXY-PASSWORD Attribute.
- Proxy Password (`-proxyPassword`) in the *Progress Startup Command and Parameter Reference*.
- Proxy Host (`-proxyhost`) in the *Progress Startup Command and Parameter Reference*.
- Proxy Port (`-proxyport`) in the *Progress Startup Command and Parameter Reference*.
- The "Proxy Authentication For WebClient" section in Appendix G, "Progress WebClient" of this bulletin.
- *Progress on the Web*

Quoter Function

Interface	OS	Speedscript
All	All	Yes

Converts the specified data type to CHARACTER and encloses the results in quotes when necessary.

The QUOTER function is intended for use in QUERY-PREPARE where a character predicate must be created from a concatenated list of string variables to form a WHERE clause. In order to process variables, screen values, and input values so that they are suitable for a query WHERE clause, it is often necessary to enclose them in quotes. For example, European-format decimals and character variables must always be enclosed in quotes. You can use the Quoter function to meet that requirement.

The Quoter function has the following syntax:

SYNTAX

```
QUOTER (<expression> [, <quote-char> [ , <null-string> ]])
```

expression

An *expression* in the data type that you want to convert to character and enclose with quotes.

quote-char

Either a single or double quote, enclosed in the opposite: """ or "'". The default is double quote. Passing ? for this argument results in double quotes.

null-string

The string you want for UNKNOWN: the word NULL or "" for example. The default is an unquoted ? character.

For example, the following:

```
 DEFINE VARIABLE mychar As CHARACTER INITIAL "Lift Line Skiing".
     …..
     qhandle:QUERY-PREPARE( "FOR EACH customer WHERE name = "  +
QUOTER(mychar))
```

Would produce this prepare-string:

```
FOR EACH customer WHERE name = "Lift Line Skiing".
```

NOTES

- To address the situation where an UNKNOWN value in a list of concatenated strings could cause the entire string to be UNKNOWN and the QUERY-PREPARE to fail, the QUOTER function does **not** return UNKNOWN if the *<expression>* argument is UNKNOWN. Instead, it returns a known character value consisting of an UNQUOTED question-mark, by default, or the 3rd argument, if it is present.

  Also, in this situation, a quoted question-mark is not used because it is interpreted as string data in a WHERE clause. After the concatenation is complete, Progress supplies a normal question mark.

  For example, the following:

```
 DEFINE VARIABLE mychar As CHARACTER.
     …..
     mychar = ?.
     qhandle:QUERY-PREPARE( "FOR EACH customer WHERE name = "  +
QUOTER(mychar)).
```

  Would produce this prepare-string:

```
FOR EACH customer WHERE name = ?.
```

  However, giving the 3rd parameter as `"NULL"` produces `NULL` rather than `?`.

- For noncharacter data types, if <*expression*> is of type DECIMAL, INTEGER, DATE , and so on, the following occurs:

  – The <*expression*> is converted to character and enclosed in quotes. The conversion is similar to the EXPORT format. DATE types, however, always have the 4-digit year.

  – Data types with no DISPLAY format like MEMPTR and LVARBINARY return the same question mark as UNKNOWN.

  – If a data type is of type RAW, it is converted to base 64.

  For example, the following:

```
 DEFINE VARIABLE mydec as DECIMAL INITIAL 12.34.
     …..
     qhandle:QUERY-PREPARE( "FOR EACH customer WHERE balance = "  +
QUOTER(mydec)).
```

  Would produce this prepare-string:

```
FOR EACH customer WHERE balance = "12.34".
```

  This is especially important for European format decimals that look like 12,34 and would not compile in the above statement unless they are enclosed in quotes.

- If <*expression*> is of data type CHARACTER, internal quotes are doubled. If the first and last byte are already quotes, then it is assumed that the quoting has already been done, and no further quotes are applied.

RETURN-VALUE Attribute

The value returned.

NOTE: Applies only to user-defined functions, attributes, and methods.

NOTE: Supported only if the CALL-TYPE attribute is set to FUNCTION-CALL-TYPE or GET-ATTR-CALL-TYPE.

Data Type: The value of the RETURN-VALUE-DATA-TYPE attribute

Access: Readable

Applies To: CALL Object

The default is ? (the unknown value).

If you set the RETURN-VALUE-DATA-TYPE attribute to a particular data type before executing INVOKE(), the value returned is converted to that data type.

If RETURN-VALUE-DATA-TYPE is set to its default value or to the expected data type, when the dynamic invoke returns, the value returned by the user-defined function or attribute is not converted.

RETURN-VALUE-DATA-TYPE Attribute

NOTE: Applies only to user-defined functions, attributes, and methods.

When you dynamically invoke a user-defined function, get an attribute, or run a method, RETURN-VALUE-DATA-TYPE lets you do one of the following to the value returned:

- Detect its data type
- Cause it to be converted to a different data type

If you set RETURN-VALUE-DATE-TYPE to a data type other than the expected data type before executing the dynamic invoke, the value returned is automatically converted to that data type. Otherwise, when the dynamic invoke returns, RETURN-VALUE-DATA-TYPE indicates the data type of the value returned.

Data Type: CHARACTER

Access: Readable/Writable

Applies To: CALL Object

SYNTAX

```
RETURN-VALUE-DATA-TYPE [ = datatype ]
```

datatype

A CHARACTER expression indicating one of the following:

- "CHARACTER"
- "DATE"
- "LOGICAL"
- "INTEGER"
- "DECIMAL"
- "RAW"
- "HANDLE"

The default is ? (the unknown value).

SAX-PARSE() Method

Performs a single-call parse of an XML document associated with a SAX-READER object.

Return Type: No return value.

Applies To: SAX-READER Object

SYNTAX

```
SAX-PARSE ( )
```

Returns when one of the following occurs:

- The parser reaches the end of the document.
- A callback invokes RETURN ERROR.
- An error occurred because the parser could not start or could not continue parsing.

While SAX-PARSE() is running, each time the parser detects an XML element, Progress invokes the corresponding callback. Callbacks reside in a procedure file whose handle has been assigned to SAX-READER's HANDLER attribute.

NOTE: SAX-PARSE() does not have a return value. To detect if an error has occurred, add NO-ERROR to the call and when it returns, check ERROR-STATUS:ERROR.

SAX-PARSE-FIRST() Method

Initializes a progressive-scan parse of an XML document associated with a SAX-READER object.

Called to begin the parse.

Return Type: No return value.

Applies To: SAX-READER Object

SYNTAX

```
SAX-PARSE-FIRST ( )
```

To continue a progressive-scan parse, call SAX-PARSE-NEXT() repeatedly.

When SAX-PARSE-FIRST() executes, Progress invokes the StartDocument callback, if there is one.

SAX-PARSE-FIRST() raises a Progress error if the parser fails, for any reason, to start parsing the XML source.

If SAX-PARSE-FIRST() is called on a source file already being parsed with SAX-PARSE-NEXT(), the parser reinitializes the parsing process.

SAX-PARSE-FIRST() can be called with SAX-READER in any state.

NOTE: SAX-PARSE-FIRST() does not have a return value. To detect if an error has occurred, add NO-ERROR to the call and when it returns, check ERROR-STATUS:ERROR.

SAX-PARSE-NEXT() Method

Continues a progressive-scan parse of an XML document associated with a SAX-READER object.

Return Type: No return value

Applies To: SAX-READER Object

SYNTAX

```
SAX-PARSE-NEXT ( )
```

SAX-PARSE-NEXT() returns after the parser finds the next XML token in the XML source and Progress invokes the corresponding callback, if it exists.

When SAX-PARSE-NEXT() starts, an error is raised if the PARSE-STATUS attribute is anything other than SAX-RUNNING. This might occur if:

- SAX-PARSE-NEXT() is called before SAX-PARSE-FIRST() is called.
- STOP-PARSING() is called.

Eventually, SAX-PARSE-NEXT() fails to find another XML token (assuming the parse does not stop early). When this happens, Progress sets the PARSE-STATUS attribute to SAX-COMPLETE. If SAX-PARSE-NEXT() is called at that point, an error is raised.

NOTE: SAX-PARSE-NEXT() does not have a return value. To detect if an error has occurred, add NO-ERROR to the call and when it returns, check ERROR-STATUS:ERROR.

NOTE: To stop a progressive-scan parse, you can refraining from calling SAX-PARSE-NEXT(). However, when you detect that it is time to stop, you will most likely be in a callback, so you need a way to communicate that it is time to stop to the driver (where SAX-PARSE-NEXT() is called). A convenient way to do that is to call SELF:STOP-PARSING(), since that causes Progress to set the value of PARSE-STATUS to SAX-COMPLETE.

SAX-ATTRIBUTES Object

Interfaces	OS	SpeedScript
All	All	Yes

Contains information on the attributes of an XML element.

Progress creates the SAX-ATTRIBUTES object, passes it to the StartElement callback, and deletes it as soon as StartElement finishes.

An application cannot create a SAX-ATTRIBUTES object.

NOTE: If an application deletes a SAX-ATTRIBUTES object, no harm occurs.

To save the data from a SAX-ATTRIBUTES object, the application must save the individual data items.

SYNTAX

```
sax-attributes-handle [ :attribute | :method ]
```

sax-attributes-handle

A handle to a SAX-ATTRIBUTES object.

attribute

An attribute of the SAX-ATTRIBUTES object. The attributes are summarized in Table D–9.

Table D–9: Attributes Of the SAX-ATTRIBUTES Object *(1 of 2)*

Attribute	Type	Description
ADM-DATA Attribute	CHARACTER	An arbitrary string for use by the ADM only.
NUM-ITEMS Attribute	INTEGER	The number of XML attributes contained in a SAX-ATTRIBUTES object.

Table D–9: Attributes Of the SAX-ATTRIBUTES Object *(2 of 2)*

Attribute	Type	Description
PRIVATE-DATA Attribute	CHARACTER	An arbitrary string the application can use any way it wants.
TYPE Attribute	CHARACTER	The type of a SAX-ATTRIBUTES handle, which is "SAX-ATTRIBUTES."
UNIQUE-ID Attribute	INTEGER	A value guaranteed by Progress to be unique within the Progress session.

method

A method of the SAX-ATTRIBUTES object. The methods are summarized in Table D–10.

Table D–10: Methods Of the SAX-ATTRIBUTES Object *(1 of 2)*

Method	Return Type	Description
GET-INDEX-BY-NAMESPACE-NAME() Method	INTEGER	Gets the index of the attribute with the given namespace name.
GET-INDEX-BY-QNAME() Method	INTEGER	Gets the index of the attribute with the given XML qualified name.
GET-LOCALNAME-BY-INDEX() Method	CHARACTER	Gets the local (unqualified) name of the attribute at the given index.

Table D–10: Methods Of the SAX-ATTRIBUTES Object *(2 of 2)*

Method	Return Type	Description
GET-QNAME-BY-INDEX() Method	CHARACTER	Gets the XML qualified name of the attribute at the given index.
GET-TYPE-BY-INDEX() Method	CHARACTER	Gets the type of the attribute at the given index.
GET-TYPE-BY-NAMESPACE-NAME() Method	CHARACTER	Gets the type of the attribute at the given namespace name.
GET-TYPE-BY-QNAME() Method	CHARACTER	Gets the type of the attribute with the given XML qualified name.
GET-URI-BY-INDEX() Method	CHARACTER	Gets the namespace URI of the attribute at the given index.
GET-VALUE-BY-INDEX() Method	CHARACTER	Gets the value of the attribute at the given index.
GET-VALUE-BY-QNAME() Method	CHARACTER	Gets the value of the attribute with the given XML qualified name.
GET-VALUE-BY-NAMESPACE-NAME() Method	CHARACTER	Gets the value of the attribute with the given namespace name.

SAX-READER Object

Interfaces	OS	SpeedScript
All	All	Yes

Provides access to the XML parser when used with the Simple API for XML (SAX). The SAX-READER object, used with SAX, corresponds to the X-DOCUMENT object of the Document Object Model (DOM) interface, but presents a programming interface that is radically different.

SYNTAX

```
sax-reader-handle [ :attribute | :method ]
```

sax-reader-handle

A handle to a SAX-READER object.

attribute

An attribute of the SAX-READER object. Table D–11 summarizes each attribute.

Table D–11: SAX-READER Object Attributes *(1 of 2)*

Attribute	Type	Description
ADM-DATA Attribute	CHARACTER	An arbitrary string for use by the ADM only.
HANDLER Attribute	HANDLE	A handle to the procedure containing the callbacks.
LOCATOR-COLUMN-NUMBER Attribute	INTEGER	The current column in the XML source.
LOCATOR-LINE-NUMBER Attribute	INTEGER	The current line in the XML source.

Table D–11: SAX-READER Object Attributes *(2 of 2)*

Attribute	Type	Description
LOCATOR-PUBLIC-ID Attribute	CHARACTER	The public identifier of the current XML source.
LOCATOR-SYSTEM-ID Attribute	CHARACTER	The system identifier of the current XML source.
PARSE-STATUS Attribute	INTEGER	The current status of the parse. Valid values are: SAX-UNINITIALIZED SAX-RUNNING SAX-COMPLETE SAX-PARSER-ERROR
PROXY-PASSWORD Attribute	CHARACTER	An arbitrary string.
SERVER Attribute	CHARACTER	The path of a DTD associated with the XML document to be parsed.
SUPPRESS-NAMESPACE-PROCESSING Attribute	LOGICAL	Whether the parser performs namespace processing.
UNIQUE-ID Attribute	CHARACTER	The type of a SAX-READER handle, which is "SAX-READER."
UNIQUE-ID Attribute	INTEGER	A value guaranteed by Progress to be unique within the Progress session.
VALIDATION-ENABLED Attribute	LOGICAL	Whether the parser performs validation.

method

A method of the SAX-READER object. Table D–12 summarizes each method.

Table D–12: Methods Of the SAX-READER Object

Method	Return Type	Description
SAX-PARSE() Method	None. (Check ERROR-STATUS: ERROR.)	Performs a single-call parse of an XML document.
SAX-PARSE-FIRST() Method	None. (Check ERROR-STATUS: ERROR.)	Initializes a progressive-scan parse of an XML document. Called once at the beginning of the parse.
SAX-PARSE-NEXT() Method	None. (Check ERROR-STATUS: ERROR.)	Continues a progressive-scan parse of the XML document. Called repeatedly to continue the parse.
SET-INPUT-SOURCE() Method	LOGICAL	Defines the source of the XML document to be parsed.
STOP-PARSING() Method	LOGICAL	Causes the parser to stop parsing the XML document.

SEE ALSO

CREATE SAX-READER Statement, SAX-ATTRIBUTES Object.

SCHEMA-PATH Attribute

A delimiter-separated list of directory paths used to locate the XML Document Type Definition (DTD) associated with a particular XML document. It is searched if the XML document contains a relative path to the DTD.

Data Type: CHARACTER

Access: Readable/Writable

Applies To: SAX-READER Object, X-DOCUMENT Object

The default is ? (the unknown value).

To separate directory paths, use a comma, or a delimiter character that your operating system recognizes.

To separate directory names and filenames within a path, use the UNIX format (forward slashes) or the standard format for your operating system.

SCHEMA-PATH can include HTTP URLs, but they might slow your application since accessing a DTD over the Web might take significantly longer than doing so over a file network.

SCHEMA-PATH cannot contain Progress procedure libraries (`.pl` files).

File pathnames cannot contain embedded commas. Progress replaces delimiter characters with commas, so the resulting SCHEMA-PATH can be accessed using the ENTRY() function.

NOTE: When Progress searches for the DTD, Progress concatenates each entry in SCHEMA-PATH with SYSTEM-ID (an attribute in DOM and a parameter of the ResolveEntity callback in SAX), which indicates the system ID of the external DTD. SYSTEM-ID might be as simple as a filename or as complex as a relative path.

For more information on accessing XML documents using the Document Object Model (DOM) or Simple API for XML (SAX) interfaces, see the *Progress External Program Interfaces* book.

SERVER Attribute

A handle to a Progress AppServer containing logic you want to invoke dynamically.

Data Type: HANDLE

Access: Readable/Writable

Applies To: CALL Object

SYNTAX

```
SERVER [ = handle-expression ]
```

handle-expression

A HANDLE expression indicating a handle to an AppServer.

The default is ? (the unknown value).

To dynamically invoke logic that resides on an AppServer, set SERVER to the handle of the AppServer.

NOTE: Do not confuse SERVER and IN-HANDLE. SERVER indicates a handle to an AppServer containing logic you want to invoke dynamically. IN-HANDLE indicates

a handle to a running persistent procedure containing logic (internal procedures and user-defined functions) you want to invoke dynamically.

SET-INPUT-SOURCE() Method

Specifies the source of the XML to be parsed by the SAX-READER.

Return Type: LOGICAL

Applies To: SAX-READER Object

SYNTAX

```
SET-INPUT-SOURCE ( mode, { file | memptr | WEB-CONTEXT } )
```

mode

A CHARACTER expression evaluating to "FILE," "MEMPTR," or "HANDLE," indicating whether the XML source is a file, a MEMPTR, or the WEB-CONTEXT handle.

file

A CHARACTER expression that indicates the name of a file. This can be a relative pathname, an absolute pathname, or an HTTP URL.

memptr

A MEMPTR variable that contains the loaded XML text. The size of the MEMPTR variable must match the size of the XML text.

WEB-CONTEXT

The WEB-CONTEXT system handle. In a WebSpeed application, this directs SAX-READER to get the XML source from WEB-CONTEXT.

The following checks are not performed by SET-INPUT-SOURCE() but are performed by SAX-PARSE() at run time:

- Whether a file exists and is accessible
- Whether a MEMPTR is usable
- Whether WEB-CONTEXT's XML attribute is YES.

If WEB-CONTEXT is used outside of a WebSpeed environment, a run-time error is raised.

SET-PARAMETER() Method

Lets you specify a parameter.

Return Type: LOGICAL

Applies To: CALL Object Handle

SYNTAX

```
SET-PARAMETER(parameter-number, data-type, iomode, parameter-value)
```

parameter-number

An INTEGER expression indicating the order of the parameter. Use 1 for the first parameter, 2 for the second parameter, etc.

data-type

A CHARACTER expression indicating the data type of the parameter and evaluating to one of the following:

- "CHARACTER"
- "DATE"
- "LOGICAL"
- "INTEGER"
- "DECIMAL"
- "RAW"
- "HANDLE"

- "MEMPTR"
- "ROWID"
- "TABLE-HANDLE"

NOTE: To manage BUFFER and TABLE parameters, use HANDLE parameters.

For each parameter, the data type specified by the caller and the callee must be compatible.

iomode

A CHARACTER expression indicating the mode of the parameter and evaluating to one of the following:

- "INPUT"
- "OUTPUT"
- "INPUT-OUTPUT"
- "OUTPUT-APPEND"

For each parameter, the mode specified by the caller and the callee must match.

For parameters of attributes, specify "INPUT."

For TABLE-HANDLE output parameters whose output is to be appended to the table, and in no other case, specify "OUTPUT-APPEND."

parameter-value

An expression whose type is compatible with *data-type*.

NOTE: TABLE-HANDLE can be a handle to a static or dynamic TEMP-TABLE.

If *iomode* is "INPUT" or "INPUT-OUTPUT," if the data type of *parameter-value* does not agree with the data type passed, *parameter-value* will automatically be converted to the data type passed during SET-PARAMETER() processing.

If iomode is "OUTPUT" or "INPUT-OUTPUT," each of the following must be true:

- *parameter-value* must represent a program variable or a NO-UNDO TEMP-TABLE field (perhaps with an array reference).

- *parameter-value* must still be in scope and must still be valid when the dynamic invoke is executed.

  If the output value from the called procedure does not have a data type that matches the data type passed, the output value will be converted to the data type passed.

If iomode is "OUTPUT," *parameter-value* is ignored if the ASYNCHRONOUS attribute is TRUE.

SET-SOCKET-OPTION( ) Method

Sets the specified socket option. TCP supports a number of socket options. Please refer to TCP documentation for a description of these options.

Return Type: LOGICAL

Applies To: Socket System Handle

SYNTAX

```
SET-SOCKET-OPTION( name , arguments )
```

name

A character expression which indicates the name of the socket option to be set.

arguments

A character expression that contains a comma separated list of arguments specific for the option.

Table D–13 describes the options Progress supports.

Table D–13: Options For SET-SOCKET-OPTION() *(1 of 2)*

Option	Description
TCP-NODELAY	An *enable indicator*, which is either TRUE or FALSE.
SO-LINGER	Two comma separated values: 1. The *onoff indicator*, which is either TRUE or FALSE. 2. The *linger time*. If the onoff indicator is FALSE, the linger time does not need to be provided.
SO-KEEPALIVE	Sets the TCP socket option SO_KEEPALIVE Set *arguments* to TRUE to turn this option on or to FALSE to turn it off.
SO-REUSEADDR	Sets the TCP socket option SO_REUSEADDR. Set *arguments* to TRUE to turn this option on or to FALSE to turn it off.

Table D–13: Options For SET-SOCKET-OPTION() *(2 of 2)*

Option	Description
SO-RCVBUF SO-SNDBUF	Sets the TCP socket option SO_RCVBUF or SO_SNDBUF. Set *arguments* to the desired size of the buffer. **NOTE:** Depending on your platform, the value you supply might be increased to the platform's minimum buffer size, decreased to the platform's maximum buffer size, or rounded up to the next multiple of the platform's segment size. For more information, see your platform's documentation.
SO-RCVTIMEO	Sets the timeout length — that is, the number of seconds the socket waits to receive data before timing out. Set *arguments* to the desired timeout value in seconds. If a timeout occurs, READ() returns TRUE and the value of BYTES-READ is zero. This is true whether the READ() mode is READ-AVAILABLE or READ-EXACT-NUM. For more information on the interaction of READ(), the READ() mode, and SO_RCVTIMEO, see the "Sockets" chapter of the *Progress External Program Interfaces* manual.

The SET-SOCKET-OPTION( ) method returns TRUE if setting the option succeeded and returns FALSE otherwise. An error can occur if:

- *name* is not a Progress supported socket option.
- The arguments supplied for the option are not valid.
- The SET-SOCKET-OPTION() operation fails.

STOP-PARSING() Method

Causes the parser to stop parsing the XML source. This lets an application search for particular data, then abort the parse as soon as the data are found.

Return Type: LOGICAL

Applies To: SAX-READER Object

SYNTAX

```
STOP-PARSING ( )
```

STOP-PARSING() can stop a parse started by SAX-PARSE(), SAX-PARSE-FIRST() or SAX-PARSE-NEXT(). That is, the parse can be single call or multiple scan.

STOP-PARSING sets SAX-READER's PARSE-STATUS attribute to SAX-COMPLETE.

Within a callback, to invoke STOP-PARSING(), use the SELF system handle, as shown in the following fragment:

```
SELF:STOP-PARSING().
```

If STOP-PARSING() is invoked in a callback or in any procedure called directly or indirectly by a callback, Progress continues to execute the callback as usual, but when the callback finishes, control returns to the next 4GL statement after the most-recently-executed SAX-PARSE(), SAX-PARSE-FIRST(), or SAX-PARSE-NEXT().

SUPPRESS-NAMESPACE-PROCESSING Attribute

Whether namespace processing is suppressed.

Data Type: LOGICAL

Access: Readable/Writable

Applies To: SAX-READER Object and X-DOCUMENT Object

FALSE, the default, indicates that namespace processing is not suppressed. TRUE indicates that namespace processing is suppressed.

For more information on accessing XML documents using the Document Object Model (DOM) and SAX interfaces, see the *Progress External Program Interfaces* book.

TYPE Attribute

The type of a SAX-ATTRIBUTES handle. Its value is “SAX-ATTRIBUTES.”

Data Type: CHARACTER

Access: Readable

Applies To: SAX-ATTRIBUTES Object and SAX-READER Object

UNIQUE-ID Attribute

A value that Progress guarantees is unique within the Progress session.

Data Type: INTEGER

Access: Readable

Applies To: SAX-ATTRIBUTES Object and SAX-READER Object

VALIDATION-ENABLED Attribute

Whether the parser validates the XML document against the DTD.

Data Type: LOGICAL

Access: Readable/Writable

Applies To: SAX-READER Object

TRUE indicates that the parser validates the XML document against the DTD. The default is TRUE.

NOTE: If VALIDATION-ENABLED is FALSE, the parser still checks that the XML document is well formed.

VIEW-FIRST-COLUMN-ON-REOPEN Attribute

Controls whether the browse, when an OPEN Query statement is run, displays the first column in the viewport or the columns that were in the viewport before the Query was reopened.

Data Type: LOGICAL

Access: Readable/Writeble

Applies To: Browse widget

When this value is set to:

- TRUE, and the query for a browse is reopened, the browse displays the first row of data and the first column in the leftmost position.
- FALSE, and the query for a browse is reopened, the browse displays the first row of data and the columns that were displayed in the viewport before the query was reopened.

  For example, if this attribute is set to FALSE and a user had scrolled to the far-right column, the next time the browse for a query is reopened, the browse displays the first row of data and the far-right column.

The default value is FALSE.

XML-SCHEMA-PATH Attribute (WebSpeed Only)

A delimiter-separated list of directory paths for the XML Document Type Definition (DTD) associated with a particular XML document in a WebSpeed environment. Searched if the XML document contains a relative path to the DTD.

Data Type: CHARACTER

Access: Readable/Writable

Applies To: WEB-CONTEXT Handle

Almost identical to the SCHEMA-PATH attribute of the X-DOCUMENT and SAX-READER objects. For more information on SCHEMA-PATH, see the SCHEMA-PATH Attribute reference entry in this book.

What XML-SCHEMA-PATH avoids:

In WebSpeed, the first time you access the X-DOCUMENT handle or any of its attributes, you trigger a load of the document, which precedes your access. For example, if you set the SCHEMA-PATH attribute of X-DOCUMENT (of WEB-CONTEXT) before accessing X-DOCUMENT (of WEB-CONTEXT) or any of its attributes, when the document arrives, WebSpeed first loads the document, then sets SCHEMA-PATH to your value. So at load time, SCHEMA-PATH might not contain your value, which might cause WebSpeed not to find your DTD, which might cause validation of the document to fail

By contrast, if you set XML-SCHEMA-PATH instead of SCHEMA-PATH, when the document arrives, WebSpeed assigns the value of XML-SCHEMA-PATH to SCHEMA-PATH before doing anything else. As a result, WebSpeed searches for your DTD.

How to use XML-SCHEMA-PATH:

In WebSpeed, instead of accessing the SCHEMA-PATH attribute of X-DOCUMENT (of WEB-CONTEXT), access XML-SCHEMA-PATH. For example, if you want a WebSpeed application to set the DTD search path (perhaps based on a Web request) of an XML document, instead of having the application set the SCHEMA-PATH attribute of X-DOCUMENT (of WEB-CONTEXT), have the application set the XML-SCHEMA-PATH attribute (of WEB-CONTEXT).

For more information on accessing XML documents using the Document Object Model (DOM) and SAX interfaces, see the *Progress External Program Interfaces* book.

XML-SUPPRESS-NAMESPACE-PROCESSING Attribute (WebSpeed Only)

Whether to suppress namespace processing.

Data Type: LOGICAL

Access: Readable/Writeable

Applies To: WEB-CONTEXT Handle

Almost identical to the SUPPRESS-NAMESPACE-PROCESSING attribute of the X-DOCUMENT handle. For more information on SUPPRESS-NAMESPACE-PROCESSING, see the XML-SUPPRESS-NAMESPACE-PROCESSING Attribute reference entry in this book.

What XML-SUPPRESS-NAMESPACE-PROCESSING avoids:

In WebSpeed, the first time you access the X-DOCUMENT handle or any of its attributes, either directly or indirectly, you trigger a load of the document, which precedes your access. For example, if you set the SUPPRESS-NAMESPACE-PROCESSING attribute of X-DOCUMENT (of WEB-CONTEXT) before accessing X-DOCUMENT (of WEB-CONTEXT) or any of its attributes, when the document arrives, WebSpeed first loads the document, then sets SUPPRESS-NAMESPACE-PROCESSING to your value. So at load time, SUPPRESS-NAMESPACE-PROCESSING might not be set to your value, which might cause document validation to fail.

By contrast, if you set XML-SUPPRESS-NAMESPACE-PROCESSING instead of SUPPRESS-NAMESPACE-PROCESSING, when the document arrives, WebSpeed assigns the value of XML-SUPPRESS-NAMESPACE-PROCESSING to SUPPRESS-NAMESPACE-PROCESSING before doing anything else. Even if this triggers a load of the document, SUPPRESS-NAMESPACE-PROCESSING already contains your value, so WebSpeed is able to validate your document.

How to use XML-SUPPRESS-NAMESPACE-PROCESSING:

In WebSpeed, instead of accessing SUPPRESS-NAMESPACE-PROCESSING of X-DOCUMENT (of WEB-CONTEXT), access XML-SUPPRESS-NAMESPACE-PROCESSING (of WEB-CONTEXT). For example, if you want a WebSpeed application to turn namespace processing off (perhaps in response to a Web request), instead of having the application assign FALSE to SUPPRESS-NAMESPACE-PROCESSING of X-DOCUMENT (of WEB-CONTEXT) have the application assign FALSE to XML-SUPPRESS-NAMESPACE-PROCESSING (of WEB-CONTEXT).

For more information on accessing XML documents using the SAX and X-DOCUMENT interfaces, see the *Progress External Program Interfaces* book.

E

SQL Explorer

This appendix documents changes made to SQL Explorer and provides information about changes made in Version 9.1 Through Version 9.1C.

E.1 Changes Made In Version 9.1 Through Version 9.1C

SQL Explorer is a Java-based tool you can use to execute SQL-92 statements interactively.

Progress Version 9.1 includes the following SQL Explorer enhancements:

- SQL Explorer Tool Updates
- Additional SQL Explorer Properties
- New Properties For GUI and Character Interfaces
- Properties For the Character Interface Only

E.1.1 SQL Explorer Tool Updates

This section describes the new and changed options available in the SQL Explorer tool.

SQL Explorer Toolbar Icons

Icons on the SQL Explorer toolbar that are now consistent with the MS-Windows industry standard.

Changes To the File Menu's Save and the Save As Options

The Save As option now behaves the way the Save option functions in earlier versions. SQL Explorer presents a file dialog box, and you select a filename.

The behavior of the Save option has also changed: the text in the input pane is now saved in the previously opened file.

Options On the New View Menu

The new View menu provides the following options:

- **Enable Output Display** — Directs SQL Explorer to echo the output to the output pane. To enable the output display, choose View→ Enable Output Display. (To disable the display, choose View→ Enable Output Display again.)
- **Table Format** — Allows you to alternate between table format and ASCII format.

- **Next Result Set** — The FetchLimit property specifies the maximum number of records to be retrieved from the database. The default value for FetchLimit is 101 records. When you choose View→ Next Result Set, SQL Explorer retrieves the next set of records, up to the maximum set in FetchLimit. For example, if you set FetchLimit to 25 and execute a fetch operation, SQL Explorer returns the first 25 records to you. If you then choose View→ Next Result Set, SQL Explorer returns the next 25 records that satisfy the query. When there are no more result sets, the Next Result Set menu item is grayed.

- **Previous Result Set** — The Previous Result Set option is the inverse of the Next Result Set option. SQL Explorer returns the number of records specified in the FetchLimit property. Using a combination of View→ Next Result Set and View→ Previous Result Set, you can scroll forward and backward through the database. When there are no previous result sets, the Previous Result Set menu item is dimmed.

- **Options** — SQL Explorer properties are now grouped into four categories of operation:

  – Connection Control

  – Logging Control

  – Report Font

  – Report Formatting

SQL Explorer Tool Online Help

The SQL Explorer tool includes enhanced online HTML help.

E.1.2 Additional SQL Explorer Properties

You can modify SQL Explorer properties to change how SQL Explorer behaves. From the graphical and character interfaces, you can change the value for a specific property. This is the syntax for setting a property from the SQL Explorer prompt, using the character interface:

SYNTAX

```
SQLExplorer> @property_name_new_value ;
```

This is the syntax for setting a property in the input pane of the SQL Explorer window, using the graphical interface:

SYNTAX

```
@property_name new_value ;
```

SQL Explorer provides six new properties in this version. (For more information, see the "New Properties for GUI and Character Interfaces" section and the "Properties for the Character Interface Only" section in this chapter.)

If you do not modify the default setting for any of the new properties, SQL Explorer behaves as it did in earlier versions.

To see a list of all properties you can set, invoke the help property:

SYNTAX

```
@help  ;
```

E.1.3 New Properties For GUI and Character Interfaces

The new properties described in this section are available for both the graphical and character SQL Explorer interfaces. For consistency, the syntax diagrams are in character-mode format.

ConnectTimeout

The ConnectTimeout property allows you to specify the maximum number of seconds SQL Explorer should wait when attempting to establish a connection, before terminating the connection request. The default value is 180 seconds:

SYNTAX

```
SQLExplorer> @ConnectTimeout number_of_seconds ;
```

HasFetchLimit

The HasFetchLimit property allows you to change the default value of true to the override setting of false:

SYNTAX

```
SQLExplorer> @HasFetchLimit { true | false }  ;
```

The HasFetchLimit property is related to the FetchLimit property. Earlier versions of SQL Explorer enforces a fetch limit, and the default for the FetchLimit property remains at 101 records. Setting HasFetchLimit to false allows you to disable the fetch limit completely, without requiring you to set an artificially high FetchLimit.

EscapeTimeout

The EscapeTimeout property allows you to specify the maximum number of seconds SQL Explorer should wait before returning to the shell and executing a shell command. The default value for EscapeTimeout is 15 seconds:

SYNTAX

```
SQLExplorer> @EscapeTimeout number_of_seconds ;
```

ReportFormat

The ReportFormat property allows you to override the standard format display for output data. Setting this property to the by label format directs SQL Explorer to return data in an alternative format. A single output line in the by label format returns the record id, column name, and column value for one column in the result set:

SYNTAX

```
SQLExplorer> @ReportFormat { standard | by label } ;
```

The following example shows data from the family table in the sports2000 database, presented in standard format:

```
@FetchLimit 5 ;
@ReportFormat standard ;

select relativename, relation, coveredonbenefits, from pub.family ;

RelativeName      Relation          CoveredOnBenefits
----------------  ----------        -----------------
Jenny Koberlein   Daughter          true
Ken Koberlein     Spouse            true
Bill Andrews      Son               true
Sam Andrews       Spouse            true
Kelly Smith       Daughter          true
```

The following example shows the same data from the family table in the sports2000 database, presented in by label format:

```
@FetchLimit 5 ;
@ReportFormat by label ;

select relativename, relation, coveredonbenefits, from pub.family ;

rec              col                   value

1                RelativeName          Jenny Koberlein
1                Relation              Daughter
1                CoveredOnBenefits     true
1                RelativeName          Ken Koberlein
1                Relation              Spouse
1                CoveredOnBenefits     true
2                RelativeName          Bill Andrews
2                Relation              Son
2                CoveredOnBenefits     true
2                RelativeName          Sam Andrews
2                Relation              Spouse
2                CoveredOnBenefits     true
3                RelativeName          Kelly Smith
3                Relation              Daughter
3                CoveredOnBenefits     true
```

NOTE: To change the display format of data you have already fetched, you first set the ReportFormat property, and then fetch the data again. SQL Explorer does not reformat the display of data that has already been fetched.

E.1.4 Properties For the Character Interface Only

The properties described in this section are available through the character SQL Explorer interface on both MS-Windows and UNIX. The graphical interface does not require these properties.

Pager

The Pager property allows you to change the default setting of false to the override value of true. When Pager is set to false, all of the output data is returned to the screen. When you choose true, SQL Explorer returns the number of lines specified in PagerLimit, which has a default value of 22 lines. You can then page up and down to review the output:

SYNTAX

```
SQLExplorer> @Pager { true | false }  ;
```

PagerLimit

The PagerLimit property allows you specify how many lines SQL Explorer displays on your screen. You can set this value to the maximum available display area on your screen, or to the number of lines you choose to use for output display. A typical setting is 22 lines. For PagerLimit to take effect, you must first set the Pager property to true. If the Pager property is set to the default value of false, SQL Explorer ignores the PagerLimit property setting:

SYNTAX

```
SQLExplorer> @PagerLimit number_of_lines ;
```

F

Administration

This appendix details new administration and configuration features in Progress

- Changes and Enhancements Made In Version 9.1D
- Changes and Enhancements Made In Version 9.1C

F.1 Changes and Enhancements Made In Version 9.1D

Version 9.1D includes the following changes and enhancements:

- Progress Explorer Tool
- AdminServer Security Enhancement

F.1.1 Progress Explorer Tool

The Progress Explorer Tool has been enhanced to incorporate the following updates in Progress Version 9.1D:

- A Register with NameServer check box that allows you the option of choosing a controlling NameServer to work with the Progress AppServer, the AppServer Internet Adapter (AIA), the Progress SonicMQ Adapter, the WebSpeed Transaction Server, WebSpeed Messengers, or the Progress DataServers. Prior to Version 9.1D, these Progress components were required to register with a NameServer.

  Depending on how you configure and use the Progress product, you can make the decision about whether to specify a controlling NameServer to handle client connection requests. For example, your configuration might be simple and not require the location transparency the NameServer provides, or it might be more complex and, therefore, dependent on this and other NameServer features.

  If you choose not to register an AppServer Internet Adapter with a NameServer, then the AIA will be directly connected to an AppServer client and you will need to specify the host location and port of that AppServer client. Likewise, if you choose not to register a WebSpeed Messenger with the NameServer, then the Messenger will be directly connected to a WebSpeed Transaction Server, and you will need to specify the host location and port of that WebSpeed Transaction Server.

- Enhanced AdminServer security, such as:

  – User authentication and user authorization that the system administrator can optionally enable during the installation of Progress. This increased level of security is apparent in the Progress Explorer when you log on and again when you attempt the required connection to an AdminServer.

- Improved AdminServer log details, which allow you to see more descriptive information when a login to the Progress Explorer or a connection to the AdminServer fails. The log also contains details about AdminServer login and connection success; you can choose to limit the logging of this success information by using the new `DLogLevelSecurity` AdminServer command-line option if you are concerned about log size. For more information about this option, see the "AdminServer Logging Enhancement" section.

- A check box in the SonicMQ Adapter General properties that allows you to start an Adapter instance automatically when the AdminServer starts. There is also a new Pool Range property in the SonicMQ Adapter Server properties that you use to specify settings for the number of threads within a SonicMQ Adapter broker process: Initial number of threads to start (the default is 20), Minimum threads (the default is 10), and Maximum threads (the default is 30).

- If you are running Fathom for Management Standard Edition, you also see some Fathom-specific fields in the Progress Explorer Tool. For example, when you set database configuration properties, you can choose whether to start the database monitoring agent automatically when the database starts.

- The WebSpeed Transaction Server, AppServer, and SonicMQ Adapter broker configurations now contain user name and password combinations. If your system administrator has established that you must enter your username and password to perform AdminServer operations, you must provide them in order to start, stop, or otherwise work with the AdminServer.

For more information, see the Progress Explorer Tool online help.

F.1.2 AdminServer Security Enhancement

Progress Version 9.1D provides enhanced functionality that provides administrators with the opportunity to establish more security when it comes to working with the AdminServer. This appendix provides information about the following:

- AdminServer Security At Install
- Option To Require Authorization On the Command Line
- AdminServer Logging Enhancement
- Option To Require a Valid Username and Password

AdminServer Security At Install

Progress Version 9.1D has optional functionality that allows access to the AdminServer based on a user's membership in a group that has the appropriate privileges to perform AdminServer operations.

Checking a user's group membership consists of the following two processes:

- **Authentication** — Determines the user is valid by requiring username and password. This functionality exists prior to Progress Version 9.1D.
- **Authorization** — Once authenticated, determines whether you can or cannot use the AdminServer.

During the Progress Version 9.1D installation, administrators are asked if they wants to enable user authorization:

- If the administrator chooses **not to use** authorization, the AdminServer functionality works as it did in Progress Version 9.1C. That is, there is no group checking and no authorization of users.
- If the administrator chooses **to use** authorization, the installation prompts for the name of the group or list of groups. The default group is PSCAdmin. The AdminServer is then going to require authorization and authentication for all operations it performs, including startup and shutdown.

NOTE: When you install the AdminServer, by default, it is started up using a default account called LocalSystem. The AdminServer Authorization dialog box also has a username and password option, that, if selected, changes the LocalSystem to a specific username and password.

Groups are set up in the operating system, outside of the Progress environment; however, an administrator using Progress Version 9.1D can also set up groups in a minimal fashion (locally only) during the Progress install. It is up to the administrator to determine who belongs in which particular group. If, after the Progress installation, a user attempts to perform an operation and does not belong to a group with that privilege, users are informed that they are not authorized to perform that operation and will be referred to the system administrator for assistance.

NOTE: Determining group membership is up to the administrator and is based on a variety of factors that differ from company to company, such as company policy, operating system, version number, procedures, etc.

Option To Require Authorization On the Command Line

If the administrator accepts the default installation and does not choose to use authorization, authorization can optionally be selected when starting up the AdminServer. The new command-line option for authorization with the AdminServer is AdminGroup (`-admingroup`) has the following syntax:

SYNTAX

```
-admingroup group[:group...]
```

For the AdminGroup startup parameter, there must be a minimum of one group. If multiple groups are listed, they are separated with a colon. The AdminServer will not start unless a minimum of one group exists. To perform AdminServer functions, the user has to be a valid account in one of the groups.

The following lists the user group authorization platform support:

- On UNIX
  - Any local or NIS group name
  - Supports subgroups
- On Windows
  - Any local or Domain group name
  - Supports global groups as members of local groups

AdminServer Logging Enhancement

There are now logging entries specifically related to user authentication and authorization. The log now lists both successful and failed operations in the following format:

[date][level]["security"] *UserName:UserSuppliedPwd:GroupInfo:Text*

The following describes the fields in a security entry.

- **Date** — The existing Logging tool will automatically insert the current date using the existing AdminServer log format.

- **Level** — The possible levels are 1 through 5, in compliance with the existing AdminServer log conventions. The security entry will use only the following levels:
  - 0: This level indicates an internal error.
  - 2: This level indicates an error condition and explains why the client was not authenticated or authorized.
  - 3: This level indicates success and is used for tracking purposes.
- **"security"** — This is a text constant that Progress specifies in order to simply log file scanning tools, so that an automated parser can easily identify security events.
- ***UserName*** — This field contains the user account being authenticated to the AdminServer. This field might indicate e "no-user" if the authentication and/or authorization operation failed before the authentication portion could take place. On Windows systems only, the *UserName* might be in the form [*domain*\]*UserName* where domain is the result of an account lookup operation when the user has not specified a fully qualified user account.
- ***UserSuppliedPwd*** — This field indicates whether the password being validated for the user account is one of the three following possible conditions:
  - Y: A password supplied by the user.
  - N: A password supplied by the single sign-on password generator.
  - X: The password has not yet been validated.
- ***GroupInfo*** — This field contains group authorization information. When the AdminServer initializes, it validates that a minimum of one group is accessible before allowing startup. In this instance, the field will contain the list of available groups and unavailable groups. In unavailable groups are designated with enclosing braces. The format is “group, group...;{unavailablegroup,unavailablegroup...}. On Windows only, the list of available groups might have the Windows domain prefixed in square brackets to indicate where the group name lookup operation found the entry.

When a security entry is made for an authentication or authorization operation, it can contain:

- **No Group Checking.** This indicates that the AdminServer started without the -admingroup option and no group authorization took place.
- **GroupName**. This indicates that a single group name was successfully authorized for the user with a success message logged.
- **GroupNames.** This indicates the group names that the user failed to be authorized in when the failure message was logged.

- ***Text*** — This field contains one of the messages that further explains the success or failure. The possible text messages follow:
  - User is not authenticated.
  - User is authenticated and authorized.
  - User is not authorized.
  - Failed to find the admingroup(s).
  - Failed to find the admingroup, not a valid group list.
  - Failed to find the admingroup, please provide a valid group list.
  - User password is not valid.
  - System generated password has expired.
  - Error, system generated password is not valid, user and host are valid.
  - Valid group list.

The default behavior for logging is that both success and failure events will be logged.

There is an AdminServer command-line option for JVMARGS that is called `DLogLevelSecurity`, that, when set, determines the type of logging that the AdminServer log file captures. The syntax for JVMARGS is as follows:

SYNTAX

```
JVMARGS="$JVMARGS -DLogLevelSecurity={2|3}"
```

Setting -DLogLevelSecurity= 2 stops successful logins from being logged.

Setting -DLogLevelSecurity=3 logs failures and successes.

Option To Require a Valid Username and Password

In Progress Version 9.1D, a user can require that when users are starting servers of the AdminServer (AppServer, SonicMQ, and WebSpeed) the ubroker.properties file must provide a valid username and password. This enhanced authentication for starting the AppServer, WebSpeed, and SonicMQ Adapter uses the uboker.properties file hierarchy to find usernames and passwords. A new Progress Explorer password field (which Progress recommends as the preferred method for updating the ubroker.properties file) can be set to supply the username's password.

The new command-line option that tells the AppServer, WebSpeed, and SonicMQ to require a username and password from the ubroker.properties file is Require Username (-requireusername). Progress Version 9.1D still uses the manual password field generator if you do not use the Progress Explorer. The user runs *<install-dir>*\bin\genpassword. This gives the user an obfuscated password that the user can enter into the Progress Explorer. Alternately, but not recommended, the user can cut and paste this password into the ubroker.properties file.

The Require Username syntax is as follows:

SYNTAX

```
-requireusername
```

F.2 Changes and Enhancements Made In Version 9.1C

F.2.1 Progress Splash Screen

The Progress GUI client and the WebClient now display a splash screen during startup. The screen is a blue rectangle bearing the words "Powered by Progress Software". It disappears when the Progress application starts running, usually when the user interface becomes visible.

Disabling the Splash Screen

If you do not want the splash screen to display, you can turn this feature off. Add the line, nosplash=yes, to the startup section of Progress' initialization file (progress.ini by default).

G

Progress WebClient

This appendix addresses the following Progress WebClient features available in Progress Version 9.1D:

- Progress Version 9.1D WebClient Installer Process
- Proxy Authentication For WebClient

G.1 Progress Version 9.1D WebClient Installer Process

In Progress Version 9.1D, the patch procedure and user interface associated with the existing WebClient Installer program have been updated to provide end users a clearer, more direct means to install and confirm the installation of a WebClient patch during the WebClient installation process.

WebClient end users (that is, customers who purchase the WebClient product from an ISV) employ the WebClient Installer to install the WebClient product. The WebClient installer is an InstallShield Professional V6 standard installation program.

The information in the following sections compare the former Progress Version 9.1C functionality of the WebClient Installer with the Progress Version 9.1D functionality.

G.1.1 Version 9.1C Functionality

The WebClient Installer supports two operating modes:

- Installation mode, which installs a new WebClient product.
- Maintenance mode, which supports three maintenance activities from the same Welcome screen as appears in Figure G–1.

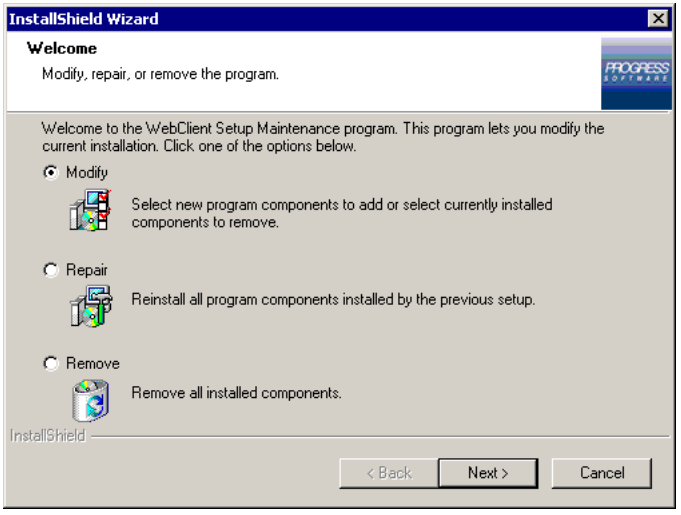


Figure G–1: Welcome Dialog Box - Maintenance Mode

Figure G–1 provides three maintenance options: Modify, Repair, and Remove. To apply a WebClient patch using this dialog box, the end user must select the Modify option. However, it is not clear that the end user must select the Modify option to apply the patch; neither the radio button label nor its associated text explanation adequately identifies this option as being specifically related to a WebClient install.

G.1.2 Version 9.1D Functionality

The WebClient installer in Version 9.1D supports three operating modes:

- Installation mode, which installs a new WebClient product.
- Maintenance mode, which supports three specific maintenance activities: Modify, Repair, and Remove to modify, repair, or remove an existing install, respectively. This is the same screen that appears in Figure G–1.
- Patch mode, a new mode dedicated to supporting the installation of a WebClient patch. This new patch mode provides three separate screens dedicated to installing a patch.

Patch Mode Dialog Boxes

Three new dialog boxes have been added to the WebClient Installer to support installing a patch. These dialog boxes automatically display when the patch trigger mechanism, a file named patchmode, is detected by the installer. Figure G–2 shows the InstallShield Wizard Welcome dialog box for the WebClient Patch Install program.

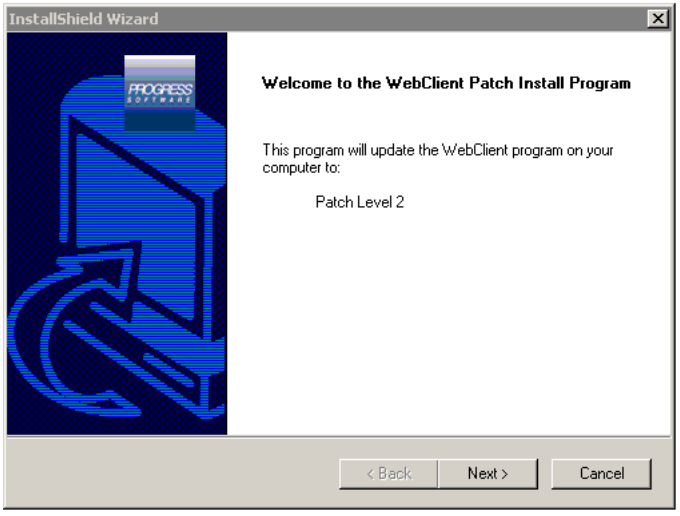


Figure G–2: Welcome To the WebClient Patch Install Program Dialog Box

This initial screen displays patch level update information. Select Next to advance the patch installation to the SetUp Status dialog box shown in Figure G–3.

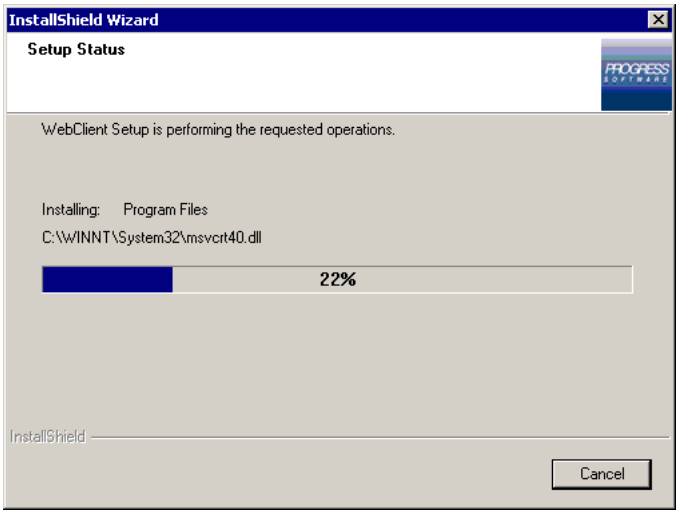


Figure G–3: SetUp Status Dialog Box

The SetUp Status screen displays a status bar to indicate that the patch install is occurring as shown in Figure G–4.

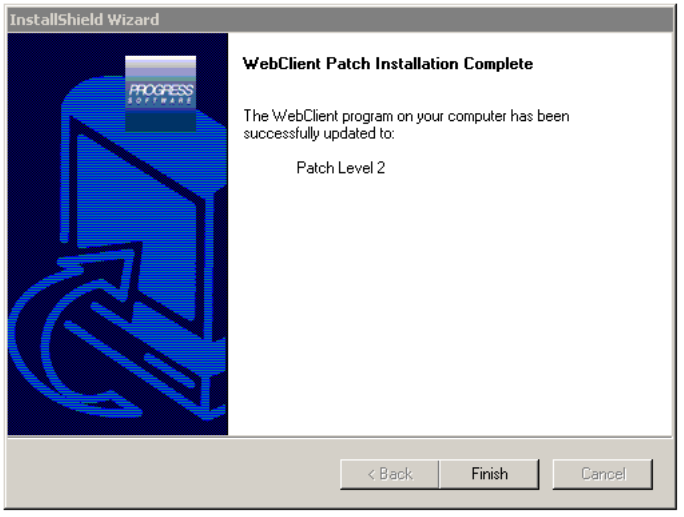


Figure G–4: WebClient Patch Installation Complete Dialog Box

This final dialog box indicates that you have completed the patch install. Select Finish to exit the InstallShield Wizard.

NOTE: If you incorrectly try to reinstall the same patch that has already been installed, the screen in Figure G–5 opens.

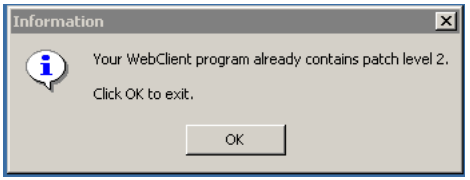


Figure G–5: Information Dialog Box

G.2 Proxy Authentication For WebClient

When WebClient downloads an Application Configuration (`.prowcapp`) file or a component (`.cab`) file from a Web server, the user is prompted for Proxy server authentication information when a proxy authentication error status is detected. The authentication information is cached in encrypted form in the registry.

The user has the option of caching the information persistently. If the user chooses to persistently cache the information, WebClient Initializer retrieves the Proxy server authentication information from the registry for subsequent downloads and passes the cached information to WebClient using `-proxyUserid` and `-proxyPassword`.

If values for `-proxyhost` and `-proxyport` are not specified in the registry, WebClient Initializer does not pass `-proxyPassword` to WebClient, even if the authentication information is cached.

For more information, see *Progress Startup Command and Parameter Reference*.

H

SQL-92

This appendix documents changes made to SQL-92 and provides information about:

- Changes Made In Version 9.1D

H.1 Changes Made In Version 9.1D

Progress Version 9.1D includes the following SQL-92 enhancements:

- Multi-threaded Progress SQL-92 Server
- Progress SQL-92 Support For Progress Sequences
- Change In Progress SQL-92 Server Locking

H.1.1 Multi-threaded Progress SQL-92 Server

With the release of Progress Version 9.1D, the Progress SQL-92 Server becomes multi-threaded. Applications written using previous versions of Progress SQL-92 now run multiple application threads on a single server. Figure H–1 shows the architecture of a multi-threaded Progress SQL-92 Server.

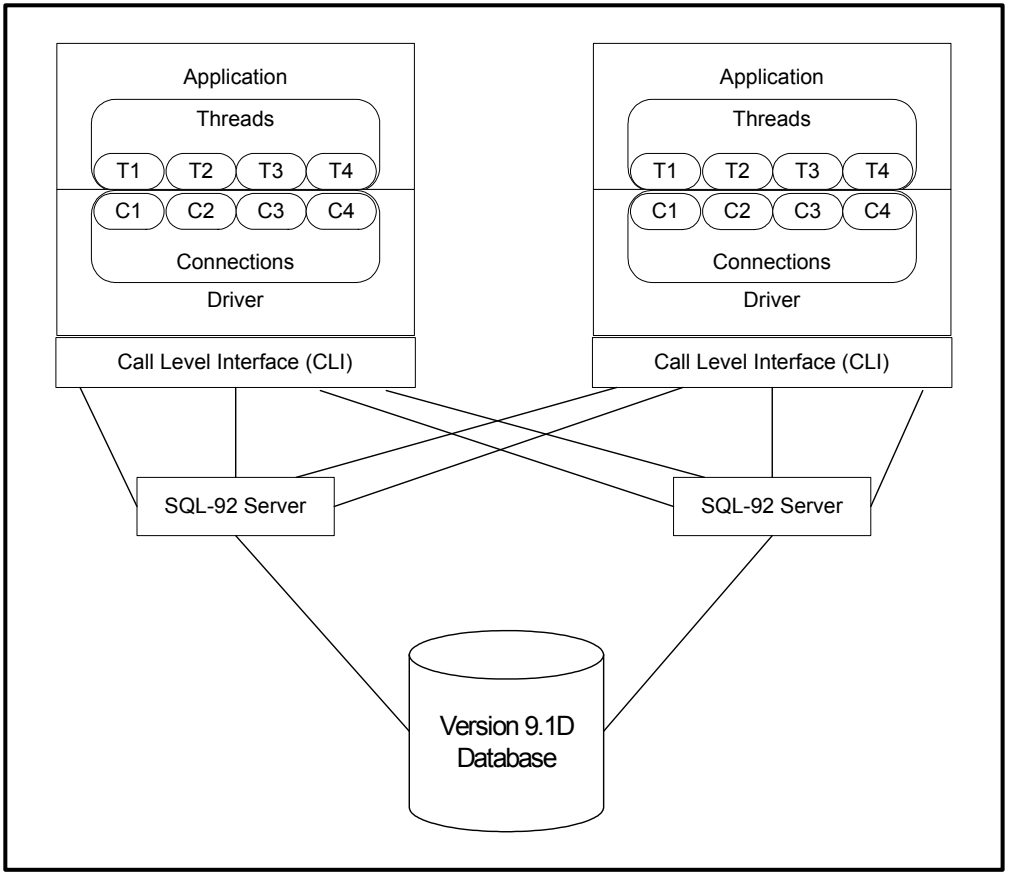


Figure H–1: Multi-threaded Progress SQL-92 Server

New Broker and Server Startup Parameters

Version 9.1D introduces three startup parameters in support of the new, multi-threaded relationship between the Progress SQL-92 server and client.

The new broker startup parameter is:

- Type of Sever to Start (`-ServerType`)

The new server startup parameters are:

- Stack Size (`-SQLStackSize`)
- Statement Cache Size (`-SQLStmtCacheSize`)

See the *Progress Startup Command and Parameter Reference* for the syntax of and detailed information for the above parameters.

H.1.2 Progress SQL-92 Support For Progress Sequences

With the release of Version 9.1D, Progress SQL-92 supports the sequence generator used in Progress 4GL applications. Sequences are database objects that generate unique integers. Because the SQL-92 sequence generator generates numbers independently of tables, the same sequence can be used for multiple tables.

In order to support the generation of sequences, the following additions were made to Progress SQL-92:

- CREATE SEQUENCE Statement
- DROP SEQUENCE Statement
- CURRVAL and NEXTVAL In a Statement
- GRANT Statement

The following sections detail the above additions to Progress SQL-92.

CREATE SEQUENCE Statement

Use the following syntax to generate a sequence:

SYNTAX

```
CREATE SEQUENCE [schema_name.]sequence_name
     [INCREMENT BY value],
     [START WITH value],
     [MAXVALUE value | NOMAXVALUE],
     [MINVALUE value | NOMINVALUE],
     [CYLCE | NOCYLCE]
```

schema_name

Specifies the schema to contain the sequence. If *schema_name* is not specified, the sequence generator creates the sequence in the current schema.

NOTE: Currently, with Version 9.1D, SQL-92 supports only the PUBLIC (PUB) schema.

sequence_name

Specifies the name of the sequence to be created.

INCREMENT BY

Specifies the interval between sequence numbers. The value can be any positive or negative integer, but cannot be 0. When the value is positive, the sequence ascends. If it is negative, the sequence descends. The default value is 1.

START WITH

Specifies the first sequence number generated. In an ascending sequence, the value must be greater than or equal to the MINVALUE. In a descending sequence, the value must be greater than or equal to the MAXVALUE. For ascending sequences, the default value is MINVALUE. For descending sequences, the default value is MAXVALUE.

MAXVALUE

Specifies the maximum value for the sequence to generate. For an ascending sequence, the default value is 2,147,483,647. For a descending sequence, the default value is -1.

NOMAXVALUE

Specifies -1 as the MAXVALUE for descending sequences and 2,147,483,647 as the MAXVALUE for ascending sequences.

MINVALUE

Specifies the minimum value the sequence can generate. For an ascending sequence, the default value is 0. For a descending sequence, the default value is -2,147,483,648.

NOMINVALUE

Specifies 0 as the MINVALUE for ascending sequences and -2,147,483,648 as the MINVALUE for descending sequences.

CYLCE

Indicates that the sequence will continue to generate values after reaching the value assigned to MAXVALUE (if sequence ascends) or MINVALUE (if sequence descends).

NOCYCLE

Indicates that the sequence cannot generate more values after reaching the value assigned to MAXVALUE (if sequence ascends) or MINVALUE (if sequence descends). The SQL-92 sequence generator uses NOCYCLE as the default if CYCLE is not specified.

In the following example, a sequence is used to generate unique customer numbers when a new customer is inserted into the table pub.customer.

EXAMPLE

```
CREATE SEQUENCE pub.customer_sequence
     START WITH 100,
     INCREMENT BY 1,
     NOCYLCE;
```

DROP SEQUENCE Statement

The DROP SEQUENCE Statement removes a sequence from a schema. You must have Database Administration privileges to remove a sequence.

Use the following syntax to remove a sequence:

SYNTAX

```
DROP SEQUENCE [schema_name.]sequence_name
```

schema_name

Specifies the schema name that contains the sequence. If this is not specified, Progress SQL-92 drops the sequence, if present, from the current schema.

NOTE: Currently, with Version 9.1D, SQL-92 supports only the PUBLIC (PUB) schema.

sequence_name

Specifies the sequence to be dropped.

In the following example, a sequence named customer is dropped from the public (PUB) schema.

EXAMPLE

```
DROP SEQUENCE pub.customer
```

CURRVAL and NEXTVAL In a Statement

Sequences contain two methods: CURRVAL and NEXTVAL. As detailed in the "CREATE SEQUENCE Statement" section, when you create a sequence you can define its initial value and the increment between its values. CURRVAL returns the current value of the sequence and NEXTVAL returns the sequence's next value. References to NEXTVAL increment the sequence value by the defined increment and return the new value. References to CURRVAL always return the sequence's current value, which is the value returned by the last reference to NEXTVAL.

Use the following syntax to reference the current value of a sequence:

SYNTAX

```
schema.sequence.CURRVAL
```

schema

Specifies the schema that contains the sequence. To refer to the current value of a sequence in the schema of another user, you must have SELECT object privilege on the sequence.

NOTE: Currently, with Version 9.1D, SQL-92 supports only the PUBLIC (PUB) schema.

sequence

Specifies the name of the sequence whose current value you want.

Use the following syntax to reference the next value of a sequence:

SYNTAX

```
schema.sequence.NEXTVAL
```

schema

Specifies the schema that contains the sequence. To refer to the next value of a sequence in the schema of another user, you must have SELECT object privilege on the sequence.

NOTE: Currently, with Version 9.1D, SQL-92 supports only the PUBLIC (PUB) schema.

sequence

Specifies the name of the sequence whose next value you want.

NOTE: A statement referencing NEXTVAL for a noncycling sequence returns an error after reaching the maximum value. For more information on cycling and noncycling sequences, see the "CREATE SEQUENCE Statement" section.

Use CURRVAL and NEXTVAL in the:

- SELECT list of a SELECT statement not contained in a subquery or view.
- SELECT list of a subquery in an INSERT statement.
- VALUES clause of an INSERT statement.
- SET clause of an UPDATE statement.

CURRVAL and NEXTVAL cannot be used in:

- A query of a view.
- A SELECT statement with a GROUP BY clause that references a sequence.
- A SELECT statement with an ORDER BY clause that references a sequence.
- A SELECT statement that is combined with another SELECT statement with the UNION, INTERSECT, or MINUS set operator.
- The WHERE clause of a SELECT or UPDATE statement.
- The DEFAULT value of a column in a CREATE TABLE or ALTER TABLE statement.
- The condition of a CHECK constraint.

EXAMPLES

In the following example, the Progress SQL-92 sequence generator returns the current value of the customer sequence:

```
SELECT customer_sequence.CURRVAL FROM pub.customer;
```

NOTE: Sequences are not associated with tables. This example provides reference to a table in order to conform to correct SQL-92 syntax.

In the following example, the sequence generator increments the customer sequence and uses its value for a new customer inserted into the table pub.customer:

```
INSERT INTO pub.customer VALUES
(customer_sequence.NEXTVAL,'USA','BackCountry Equipment','Sugar Hill
Road','12A','Franconia','NH','03242','Dan Egan','603-762-2121','Kirsten
Ulmner', 10000.00, 500.00,'net 10', 0,'contact monthly');
```

GRANT Statement

Use the GRANT statement to assign privileges for accessing sequences. As database administrator, you can assign SELECT and UPDATE privileges to specific sequence objects.

Use the following syntax to assign sequence privileges:

SYNTAX

```
GRANT [SELECT | UPDATE]
ON SEQUENCE schema.sequence
TO user_name [,user_name]...
```

SELECT

Allows specified user to read data from the sequence

UPDATE

Allows specified user to modify data for the sequence

EXAMPLE

In this example, the sequence generator grants user slsadmin the ability to modify the customer number sequence.

```
GRANT UPDATE
   ON SEQUENCE pub.customer_sequence
   TO slsadmin;
```

H.1.3 Change In Progress SQL-92 Server Locking

In versions prior to Version 9.1D, the Progress SQL-92 Server locks at the database connection level. This means that schema changes can only be performed when no users are connected to the database. With the release of Version 9.1D, the Progress SQL-92 Server locks at the transaction level. This new level of locking allows the DBA to modify database schema while there are active connections, as long as the users are not in active transactions.

Database Tools

Version 9.1D includes the following enhancement to the database tools:

- Adjusting the SQL Width Property Setting

I.1 Adjusting the SQL Width Property Setting

The SQL Field Width property allows the 4GL client to modify the field width so that the SQL-92 client will have the proper size to work with. If you do not adjust the setting, the SQL-92 client uses the default width property.

To adjust the field width in GUI mode, make sure you are connected to a database and then follow these steps:

1 ♦ From the Windows Desktop, choose Start→ Programs→ Progress→ Data Dictionary. The Data Dictionary appears.

2 ♦ Select a table.

3 ♦ Choose Options→ SQL Properties→ Adjust Field Width from the menu bar. Progress displays the Adjust Field Width dialog box.

4 ♦ Select the name of the field whose width you want to adjust, click its width setting, and make the adjustment.

5 ♦ Repeat Step 4 for any other fields.

6 ♦ Choose Save, and then choose OK.

To adjust the field width in character mode, make sure you are connected to a database and follow these steps:

1 ♦ Access the Data Dictionary.

2 ♦ Choose Schema→ SQL Properties. A table name appears in the Table Name field, which is followed by a list of additional table names.

3 ♦ Choose the table you want; the name automatically appears in the Table Name field.

4 ♦ Press **ENTER** to open the table. The width of the first field in the field name listing is highlighted.

5 ♦ Select the name of the field whose width you want to adjust, click its width setting, and make the adjustment.

6 ♦ Repeat Step 5 for any other fields.

7 ♦ Choose Save, then press **ENTER**.

8 ♦ Press **F4**, then choose Apply Changes.

J

Dynamic Invoke

This chapter tells you how to invoke code dynamically. You can dynamically execute a statement, execute a user-defined function, get an attribute, set an attribute, or execute a method.

This appendix includes the following sections:

- 4GL Elements Related To Dynamic Invoke
- Using Dynamic Invoke Only When Necessary
- Example 1 — Invoking an External Procedure Nonpersistently
- Example 2 — Running an External Procedure Persistently and One Of Its Internal Procedures
- Example 3 — Using a CALL Object Multiple Times
- Example 4 — Getting an Attribute
- Example 5 — Setting an Attribute
- Example 6 — Driving a Series Of Dynamic Invokes From a TEMP-TABLE

J.1 4GL Elements Related To Dynamic Invoke

The 4GL elements related to dynamic invoke consist of a statement, a handle, attributes, and methods.

NOTE: For more information on these 4GL elements, see the *Progress Language Reference*.

The CREATE CALL Object statement creates a CALL object, then stores a handle to it in the handle variable specified.

The CALL Object handle lets you do the following dynamically:

- Invoke an external procedure, internal procedure, or user-defined function
- Get or set an attribute
- Run a method

Table J–1 lists the attributes of the CALL Object handle.

Table J–1: Attributes Of the CALL Object Handle *(1 of 4)*

Attribute	Type	Description
ASYNCHRONOUS Attribute	LOGICAL	(Applies only when the logic to be dynamically invoked resides on a Progress AppServer.) Indicates whether you want the logic to run asynchronously.
ASYNC-REQUEST-HANDLE Attribute	HANDLE	(Applies only when ASYNCHRONOUS is TRUE.) A handle to an asynchronous-request object containing detailed information on your asynchronous request. Set by INVOKE() when ASYNCHRONOUS is TRUE.

Table J–1: Attributes Of the CALL Object Handle *(2 of 4)*

Attribute	Type	Description
CALL-NAME Attribute	CHARACTER	The name of one of the following: • A procedure or user-defined function you want to invoke dynamically • An attribute or you want to get or set dynamically • A method you want to invoke dynamically
CALL-TYPE Attribute	INTEGER	Whether you are dynamically: • Invoking a procedure • Invoking a user-defined function • Getting an attribute or invoking a method • Setting an attribute
EVENT-PROCEDURE Attribute	CHARACTER	(Applies only when ASYNCHRONOUS is TRUE.) The name of an internal procedure you want executed when a dynamic, asynchronous invoke completes. **NOTE:** You must set EVENT-PROCEDURE before you run INVOKE().

Table J–1: Attributes Of the CALL Object Handle *(3 of 4)*

Attribute	Type	Description
EVENT-PROCEDURE-CONTEXT Attribute	HANDLE	(Applies only when ASYNCHRONOUS is TRUE.) A handle to a running persistent procedure containing an internal procedure you want executed when a dynamic, asynchronous invoke completes.
IN-HANDLE Attribute	HANDLE	Contains one of the following: • A handle to a persistent procedure you just invoked dynamically • A handle to an already-running persistent procedure containing logic you want to invoke dynamically • A handle to an object whose attributes or methods you want to invoke dynamically
IS-PARAMETER-SET Attribute	LOGICAL	Whether you have already set a particular parameter.
NUM-PARAMETERS Attribute	INTEGER	The number of parameters you are passing.
PERSISTENT Attribute	LOGICAL	Whether you want an external procedure you plan to invoke dynamically to run persistently.

Table J–1: Attributes Of the CALL Object Handle *(4 of 4)*

Attribute	Type	Description
RETURN-VALUE Attribute	RETURN-VALUE-DATA-TYPE	(Applies only to user-defined functions, attributes, and methods) One of the following: • The value returned by a user-defined function • The value of an attribute • The value returned by a method
RETURN-VALUE-DATA-TYPE Attribute	CHARACTER	(Applies only to user-defined functions, attributes, and methods) The data type of the return value. If you set this before you execute INVOKE(), the return value is converted to this data type. If you do not set this before you execute INVOKE(), it indicates the return value's data type.
SERVER Attribute	HANDLE	A handle to an AppServer containing logic you want to invoke dynamically.

Table J–2 lists the methods of the CALL Object handle.

Table J–2: Methods Of the CALL Object Handle

Method	Return Type	Description
CLEAR() Method	LOGICAL	Resets each attribute of the CALL object to its default value. **NOTE:** To reset just the parameters, set the NUM-PARAMETERS attribute to zero.
INVOKE() Method	None (similar to the RUN statement)	Lets you dynamically do the following: • Invoke an external procedure, internal procedure, or user-defined function • Get or set an attribute • Invoke a method **NOTE:** INVOKE() cannot occur within an expression.
SET-PARAMETER() Method	LOGICAL	Lets you set parameters of one of the following: • A procedure or user-defined function you want to invoke dynamically • An attribute you want to get or set dynamically • A method you want to invoke dynamically

J.2 Using Dynamic Invoke Only When Necessary

The "4GL Elements Related To Dynamic Invoke" section lists the 4GL elements that dynamic invoke involves. As you can see, the list is long.

Besides involving more 4GL elements, dynamic invoke requires more lines of code. Table J–3 shows the code required to invoke an external procedure dynamically and statically.

Table J–3: Invoking hello.p Dynamically and Statically

Dynamic Invoke	Static Invoke
DEFINE VARIABLE hCall AS HANDLE. CREATE CALL hCall. /*invoke hello.p nonpersistently */ hCall:CALL-NAME = "hello.p". hCall:NUM-PARAMETERS = 1. hCall:SET-PARAMETER(1, "CHARACTER", "INPUT", "HELLO WORLD"). hCall:INVOKE. /* clean up */ DELETE OBJECT hCall.	RUN hello.p("HELLO WORLD").

As Table J–3 illustrates, performing this task using dynamic invoke requires many more lines of code than does doing it with static invoke.

NOTE: For more information on this code, see the "Example 1 — Invoking an External Procedure Nonpersistently" section.

For this reason, Progress Software Corporation recommends that you use dynamic invoke only when using static invoke is impossible.

Specifically, use dynamic invoke in the following situations:

- To invoke an internal or external procedure whose *calling sequence* (number of parameters and the data type of each) is unknown at compile time

  NOTE: But if only the **name** of the procedure is unknown at compile time, use the RUN statement with the VALUE option — and avoid dynamic invoke completely.

- To invoke a function whose calling sequence is unknown at compile time

  NOTE: But if only the **name** of the function is unknown at compile time, use the DYNAMIC-FUNCTION() function — and avoid dynamic invoke completely.

- To reference a widget attribute or method whose name is unknown at compile time

  If you already know the name of the attribute or procedure, you know its syntax, since the name implies certain syntax. And if you know the syntax, you know the calling sequence, since the syntax defines the calling sequence. And if you know the calling sequence, you can use *`widget:attribute`* or *`widget:method`* syntax — and avoid dynamic invoke completely.

J.3 Example 1 — Invoking an External Procedure Nonpersistently

The example p-diexm1.p dynamically invokes the external procedure `hello.p` nonpersistently.

p-diexm1.p

```
DEFINE VARIABLE hCall AS HANDLE.

CREATE CALL hCall.

/*invoke hello.p nonpersistently */
hCall:CALL-NAME = "hello.p".
/*sets CALL-TYPE to the default/*
hCall:CALL-TYPE = PROCEDURE-CALL-TYPE.
hCall:NUM-PARAMETERS = 1.
hCall:SET-PARAMETER( 1, "CHARACTER", "INPUT", "HELLO WORLD").
hCall:INVOKE.

/* clean up */
DELETE OBJECT hCall.
```

Example p-diexm1.p invokes the procedure file `hello.p` with a CHARACTER input parameter, the string "HELLO WORLD". The following fragment accomplishes the exact same thing statically:

```
RUN hello.p("HELLO WORLD").
```

This illustrates that dynamic invoke requires many more lines of code than does static invoke. For this reason, Progress Software Corporation recommends that you use dynamic invoke only when using static invoke is impossible.

J.4 Example 2 — Running an External Procedure Persistently and One Of Its Internal Procedures

The next example, p-diexm2.p, is a bit more complicated. It does two things:

1. Dynamically invokes the external procedure `persis.p` persistently.

2. Dynamically invokes an internal procedure of `persis.p`, internal-persis-proc, with an input parameter of type INTEGER with the value 333.

p-diexm2.p

```
DEFINE VARIABLE hCall AS HANDLE.

CREATE CALL hCall.

/*invoke persis.p persistently */
hCall:CALL-NAME = "persis.p".
/*sets CALL-TYPE to the default/*
hCall:CALL-TYPE = PROCEDURE-CALL-TYPE.
hCall:PERSISTENT = true.
HCall:INVOKE.
   /* IN-HANDLE is automatically set
      to the handle of the persistent procedure
   */

/* invoke internal-persis-proc in persis.p */
hCall:CALL-NAME = "internal-persis-proc".
/*sets CALL-TYPE to the default/*
hCall:CALL-TYPE = PROCEDURE-CALL-TYPE.
hCall:NUM-PARAMETERS = 1.
hCall:SET-PARAMETER( 1, "INTEGER", "INPUT", 333).
hCall:INVOKE.

/* clean up */
DELETE PROCEDURE hCall:IN-HANDLE.
DELETE OBJECT hCall.
```

In p-diexm2.p, when `persis.p` is invoked dynamically, the handle of the running persistent procedure is stored in CALL's IN-HANDLE attribute automatically, Then, when internal-persis-proc is invoked dynamically, Progress knows it resides in the running persistent procedure whose handle is stored in CALL's IN-HANDLE attribute.

p-diexm2.p also illustrates using a single CALL handle multiple times — in this case, twice. The next example, p-diexm3.p, demonstrates this in more detail.

J.5 Example 3 — Using a CALL Object Multiple Times

Example, p-diexm3.p further demonstrates using a CALL object multiple times by doing the following:

1. Dynamically invoking the procedure file `hello.p` with a CHARACTER input parameter, the string "HELLO WORLD"
2. Dynamically invoking the external procedure `persis.p` persistently
3. Dynamically invoking an internal procedure of `persis.p`, internal-persis-proc, with an input parameter of type INTEGER with the value 333

p-diexm3.p

```
DEFINE VARIABLE hCall AS HANDLE.

CREATE CALL hCall.

/*invoke hello.p nonpersistently */
hCall:CALL-NAME = "hello.p".
/*sets CALL-TYPE to the default/*
hCall:CALL-TYPE = PROCEDURE-CALL-TYPE.
hCall:NUM-PARAMETERS = 1.
hCall:SET-PARAMETER( 1, "CHARACTER", "INPUT", "HELLO WORLD").
hCall:INVOKE.

/* reset the CALL object handle */
hCall:CLEAR.

/*invoke persis.p persistently */
hCall:CALL-NAME = "persis.p".
/*sets CALL-TYPE to the default/*
hCall:CALL-TYPE = PROCEDURE-CALL-TYPE.

hCall:PERSISTENT = true.
HCall:INVOKE.
   /* IN-HANDLE is automatically set
      to the handle of the persistent procedure
   */

/* invoke internal-persis-proc in persis.p
HCall:CALL-NAME = "internal-persis-proc".
/*sets CALL-TYPE to the default/*
hCall:CALL-TYPE = PROCEDURE-CALL-TYPE.
hCall:NUM-PARAMETERS = 1.
hCall:SET-PARAMETER( 1, "INTEGER", "INPUT", 333).
hCall:INVOKE.

/* clean up */
DELETE PROCEDURE hCall:IN-HANDLE.
DELETE OBJECT hCall.
```

Example p-diexm3.p resets the CALL object handle by using the CLEAR() method between invoking `hello.p` (the first invoke) and invoking `persis.p` (the second invoke).

J.6 Example 4 — Getting an Attribute

Example p-diexm4.p gets the current value of the TITLE attribute of a frame.

p-diexm4.p

```
/* get title of frame */

hCall:IN-HANDLE = myframe_handle.
HCALL:CALL-TYPE = GET-ATTR-CALL-TYPE.
hCall:CALL-NAME = "TITLE".
hCall:INVOKE.
Mytitle = hCall:RETURN-VALUE.
```

In p-diexm4.p, CALL's CALL-TYPE attribute is set to GET-ATTR-CALL-TYPE. The value of the TITLE attribute is returned thorough CALL's RETURN-VALUE attribute.

J.7 Example 5 — Setting an Attribute

Example p-diexm5.p sets the SESSION handle's NUMERIC-FORMAT attribute to "european."

p-diexm5.p

```
/* set SESSION:NUMERIC-FORMAT to "european" */

hCall:IN-HANDLE = "session".
HCALL:CALL-TYPE = SET-ATTR-CALL-TYPE.
hCall:CALL-NAME = "numeric-format".
hCall:NUM-PARAMETERS = 1.
hCall:SET-PARAMETER( 1, "CHAR", "INPUT", "european").
hCall:INVOKE.
```

In contrast to Example p-diexm4.p, Example p-diexm5.p:

- Sets the CALL-TYPE attribute to SET-ATTR-CALL-TYPE
- Sets the NUM-PARAMETERS attribute to 1
- Runs the SET-PARAMETER() method make the first parameter an INPUT parameter of data type CHARACTER with the value "european"

J.8 Example 6 — Driving a Series Of Dynamic Invokes From a TEMP-TABLE

Example p-diexm6.p drives a series of dynamic invokes from a temp-table.

p-diexm6.p

```
/*suppose hRuntt is a temp-table that has one record with the
following fields:
   parm_1
   parm_2
   ...
   parm_n
   run-name
   nparms
   datatypes, extent nparms
   iomodes, extent nparms
*/

DEFINE INPUT PARAMETER TABLE-HANDLE hRuntt.

DEFINE VARIABLE hDtypes  AS HANDLE.
DEFINE VARIABLE hIOmodes AS HANDLE.
DEFINE VARIABLE hCall    AS HANDLE.
DEFINE VARIABLE i        AS INTEGER.

hDtypes  = hRuntt:BUFFER-FIELD("datatypes").
hIOmodes = hRuntt:BUFFER-FIELD("iOmodes").

hRuntt:FIND-FIRST.

CREATE CALL hCall.

hCall:CALL-NAME = hRuntt:BUFFER-FIELD("run-name"):BUFFER-VALUE.

hCall:NUM-PARAMETERS = hRuntt:BUFFER-FIELD("nparms"):BUFFER-VALUE.

FOR i = 1 to hCall:NUM-PARAMETERS.
    hCall:SET-PARAMETER( i,
           hDtypes:BUFFER-VALUE(i),
           hIOmodes:BUFFER-VALUE(i),
           hRuntt:BUFFER-FIELD(i):BUFFER-VALUE).
END.

hCall:INVOKE.
DELETE OBJECT hCall.
/* if there are output parms,
   get the values from hRuntt:BUFFER-FIELD(i)
```

The idea behind Example p-diexm6.p involves the following tasks:

1. Defining a temp-table record structure where each field represents one data item involved in dynamic invoke

2. Loading the temp-table with a row of data for each dynamic invoke to be performed

3. Reading the temp-table, performing a dynamic invoke for every record

Setting up a temp-table like this lets an application perform any number of dynamic invokes on the fly.

K

Optional NameServer Enhancement

Progress Version 9.1D provides enhanced functionality that allows users to optionally specify no NameServer. This appendix contains information about:

- Overview Of the Optional NameServer
- No NameServer On the Server
- No NameServer On the Client

K.1 Overview Of the Optional NameServer

The Optional NameServer accommodates some configurations that do not require the NameServer as well as configurations that do not use the User Datagram Protocol (UDP). In Progress versions earlier than Version 9.1D, the NameServer is required, not optional, and as a result, users had to use UDP.

Using the NameServer provides location transparency and load balancing capabilities to servers that use it, namely AppServer, WebSpeed Transaction Server, Sonic Adapter, and DataServers. (Unlike other servers, DataServers only use the NameServer when they are configured and started up using Progress Explorer.)

In versions earlier than Progress Version 9.1D, all servers that support the NameServer do so as a required component. That is, as part of configuring the component using Progress Explorer, a controlling NameServer must be specified. A client that wants to access the server must specify the appropriate connection parameters such that the request to locate the required server is sent to the specified NameServer.

There are two benefits to making the NameServer optional:

- Some configurations are simple and static. They do not need location transparency or load balancing. For these configurations, taking advantage of the optional NameServer means that they have less to configure and manage.
- Since the NameServer uses UDP and some customers have restrictions that prohibit using UDP, using the optional NameServer eliminates this dependency.

For customers who want to take advantage of the optional NameServer and choose not to use the NameServer in Progress Version 9.1D, they need to do the following:

- Configure their server in Progress to indicate that the server should not register with a NameServer. See Appendix F, “Administration” in this bulletin for more information about using Progress Explorer.
- Connect their clients directly to the server by specifying the server’s TCP/IP host and port. There is a new connection parameter, DirectConnect (`-DirectConnect`), that specifies that no NameServer should be used when the client connects to the server.

NOTE: A given server should be configured to not register with the NameServer only if all clients will not be using the NameServer to connect to the server. You should not have a server simultaneously support both clients that use the NameServer and clients that do not use the NameServer. Either all clients should use the NameServer or all clients should not use the NameServer.

K.2 No NameServer On the Server

The default behavior for Progress Version 9.1D is the same as prior versions, which means that the NameServer is used. To optionally select no NameServer in Progress Version 9.1D, the user must deselect the new "Register with NameServer" checkbox in the NameServer configuration screen of Progress Explorer for each server type. When deselected, the controlling NameServer cannot be set and the controlling NameServer drop down list is grayed out.

K.3 No NameServer On the Client

The section explains how each type of client in Progress Version 9.1D indicates that the NameServer should not be used.

K.3.1 4GL Client To AppServer

The 4GL client connects to an AppServer by specifying connection parameters through the CONNECT method executed on the SERVER handle. The connection parameters can be specified using two different syntaxes. The following section indicates how the clients connect with no NameServer for each syntax.

Explicit Specification Of Connection Parameters

In prior versions, the client can explicitly use the Host (`-H`), Server (`-S`), and AppService (`-AppService`) parameters to specify the NameServer host, port, and the requested Application Service.

In Progress Version 9.1D, to indicate that the NameServer should not be used, the DirectConnect (`-DirectConnect`) connection parameter is used to specify that the Host (`-H`) and `Server` (`-S`) parameters are interpreted as the host and port of the AppServer.

With the DirectConnect parameter, the Host and Server parameters are optional. When the Host parameter is not specified, the default value is the "localhost." When the Server is not specified, the default port is 3090.

If the DirectConnect parameter is specified, then the AppService parameter is ignored.

Connection Parameter Specified Through a URL

In versions prior to Progress Version 9.1D, a client can use URL syntax to indicate connection parameters to the AppServer by using the URL (`-URL`) connection parameter, with the following supported protocols for use with the AppServer: HTTP, HTTPS, and AppServer. Now in Progress Version 9.1D, there is an additional protocol that allows the client to connect directly to the AppServer without using the controlling NameServer. This additional protocol is AppServerDC, and it syntax is as follows:

SYNTAX

```
AppServerDC:{//host[:port]}
```

If the `AppServerDC` protocol is specified, then the *`host`* and *`port`* parameters are interpreted as the host and port of the AppServer.

With the `AppServerDC`, the *`port`* parameters is optional. When the port is not specified, the default port is 3090.

K.3.2 OpenClient To an AppServer

In Progress Version 9.1D, an OpenClient can connect to an AppServer without using the NameServer by specifying a URL with an AppServerDC protocol, as indicated in the "Connection Parameter Specified Through a URL" section.

K.3.3 WebClient To an AppServer

In Progress Version 9.1D, a WebClient can connect to an AppServer without using the NameServer by specifying a URL with an AppServerDC protocol, as indicated in the "Connection Parameter Specified Through a URL" section.

K.3.4 4GL Client To a DataServer

DataServer clients in versions prior to Progress Version 9.1D use the DataServer (`-DataService`) parameter. This parameter is still supported, or users in Progress Version 9.1D can use the DirectConnect (`-DirectConnect`) switch as specified in the "Explicit Specification Of Connection Parameters" section. If the Server (`-S`) parameter is not specified, the port defaults to 4444 for the DataServer for ODBC, 4445 for the DataServer for Oracle, and 4446 for the DataServer for Microsoft SQL Server.

K.3.5 4GL Client To Sonic Adapter

In Progress Version 9.1D, a 4GL client specifies the connection parameter to the Sonic adapter by setting the `adapterConnection` parameter that is used with the `pusubsession.p` and `ptpsession.p` procedures. The `adapterConnection` parameter supports the same syntax as that for the AppServer. The no NameServer option is supported using the same syntaxes that are specified in the "Explicit Specification Of Connection Parameters" section, except that if the `Server (-S)` parameter is not specified, it defaults to 3620.

K.3.6 WebSpeed Messenger and the AppServer Internet Adapter

The WebSpeed Messenger and the AppServer Internet Adapter (AIA) in versions prior to Progress Version 9l.1D require connecting to a server instance using a NameServer. In Progress Version 9.1D, the user can indicate no NameServer by deselecting the Connect Using NameServer check box that is in the configuration screens of WebSpeed and AIA in the Progress Explorer.

If you specify not to use the controlling NameServer, you must specify the TCP/IP host and port of the server. The host and port are only enabled if "Connect Using NameServer" is deselected. The host and the port contain the host and port of the server instance to which the Messenger or AIA instance is to connect. The default value for host is "localhost" and the default port is 3055.

L

Installation

New and revised features have been introduced to the installation and configuration processes to support Windows and UNIX platforms, respectively. This appendix describes these features in Version 9.1D and contains information about the following:

- Optional AdminServer User-group Authorization and Authentication
- Changes In the Core Installation Of Progress Messages Files
- WebClient Installation
- Supported UNIX Platform

L.1 Optional AdminServer User-group Authorization and Authentication

An optional AdminServer user-group authorization and authentication feature was added to Progress Version 9.1D. This feature allows users to specify group authorization requirements to start the Admin Service.

If you install Progress Version 9.1D on a Windows NT, Windows 2000, or Windows XP platform, the installation program prompts you to enter the required information to set this option. Alternatively, for these Windows platforms, you use this feature using command-line entry. Progress Version 9.1D supports new parameters for this purpose.

If you install Progress 9.1D on a UNIX platform, you must enter these new parameters on the command line independently of the installation process to use this feature.

NOTE: The AdminServer User-Group Authorization feature is not available on these Windows platforms: Windows 98 and Windows Millennium (Me).

For more information, see:

Manual:	Appendix F, "Administration" in this bulletin.

L.2 Changes In the Core Installation Of Progress Messages Files

Changes were made to the Core Installation of Progress Messages Files to accommodate future add-on product installations.

Specific file protection measures have been added to the Progress Core Installation process to accommodate files associated with Progress add-on products. Progress add-on products refer to categories of Progress products released independently of a point or major Progress product release. However, these products work with and depend on functionality available within the Progress Core Software product set.

NOTE: Progress add-on products will be available after Progress Version 9.1D is released. For specific information about these products, contact your Progress Sales Representative.

Prior to Progress Version 9.1D, certain functions within the Progress Core installation process have the potential to incorrectly overwrite or update a `promsgs` file in the local directory. To eliminate the possibility of this occurring, Progress Version 9.1D introduces a file protection mechanism to protect `promsgs` files and any associated files, and a file comparison technique to ensure that you always have the most recent `promsgs` files.

This Version 9.1D change ensures that users always have the most up-to-date Progress messages available for the entire Progress installation.

For more information, see:

Manual:	*Progress Installation and Configuration Guide Version 9 for UNIX* *Progress Installation and Configuration Guide Version 9 for Windows*

L.3 Updates To the Installation Process For Windows Platforms

The Version 9.1D installation process for Windows platforms has been updated to support a Silent Installation Option for the Shared Network Installation Utility (`NetSetup`)

In Progress Version 9.1D, the Shared Network Installation Utility (`NetSetup`) supports a new installation option referred to as a silent installation mode. Users can use the Installshield SILENTMODE mechanism to enable a Progress Core installation to occur in response to values supplied by a user-created file rather than by values that users interactively entered through a keyboard. A Progress Version 9.1D installation initiated using this option does not require user involvement to advance through and complete the installation process.

Manual:	*Progress Installation and Configuration Guide Version 9 for Windows*

L.4 WebClient Installation

In Progress Version 9.1D, the patch procedure and user interface associated with the existing WebClient Installer program has been updated to provide users with a clearer, more direct way to install and confirm the installation of a WebClient patch during the WebClient installation process.

For more information, see:

Manual:	Appendix G, “Progress WebClient” in this bulletin.

L.5 Supported UNIX Platform

With the release of Version 9.1D, the SUN Solaris Sparc 8 (64 Bit) UNIX platform is supported:

For more information, see:

Manual:	*Progress Installation and Configuration Guide Version 9 for UNIX*