Progress® Developer Studio for OpenEdge
Online Help
Copyright

© 2017 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

These materials and all Progress® software products are copyrighted and all rights are reserved by Progress Software Corporation. The information in these materials is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear therein. The references in these materials to specific platforms supported are subject to change.

Business Making Progress, Corticon, DataDirect (and design), DataDirect Cloud, DataDirect Connect, DataDirect Connect64, DataDirect XML Converters, DataDirect XQuery, Deliver More Than Expected, Icenium, Kendo UI, Making Software Work Together, NativeScript, OpenEdge, Powered by Progress, Progress, Progress Software Developers Network, Rollbase, SequeLink, Sitefinity (and Design), SpeedScript, Stylus Studio, TeamPulse, Telerik, Telerik (and Design), Test Studio, and WebSpeed are registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and/or other countries. AccelEvent, Analytics360, AppsAlive, AppServer, Arcade, BravePoint, BusinessEdge, DataDirect Spy, DataDirect SupportLink, DevCraft, DigitalFactory, Fiddler, Future Proof, High Performance Integration, JustCode, JustDecompile, JustMock, JustTrace, OpenAccess, ProDataSet, Progress Arcade, Progress Profiles, Progress Results, Progress RFID, Progress Software, ProVision, PSE Pro, SectorAlliance, Sitefinity, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, WebClient, and Who Makes Progress are trademarks or service marks of Progress Software Corporation and/or its subsidiaries or affiliates in the U.S. and other countries. Java is a registered trademark of Oracle and/or its affiliates. Any other marks contained herein may be trademarks of their respective owners.

Please refer to the Release Notes applicable to the particular Progress product release for any third-party acknowledgements required to be provided in the documentation associated with the Progress product.

The Release Notes can be found in the OpenEdge installation directory and online at: https://community.progress.com/technicalusers/w/openedgegeneral/1329.openedge-product-documentation-overview.aspx.

For the latest documentation updates see OpenEdge Product Documentation on Progress Communities: (https://community.progress.com/technicalusers/w/openedgegeneral/1329.openedge-product-documentation-overview.aspx).

March 2017

Last updated with new content: Release 11.7.0

Updated: 2017/03/24
# Table of Contents

**Preface**

- About this manual
- Typographical conventions

**Chapter 1: Introducing Progress Developer Studio for OpenEdge**

- Concepts
  - What's New in Release 11.7
  - New and Noteworthy in Release 11.6
  - New and Noteworthy in Release 11.5
  - New and Noteworthy in Release 11.4
  - New and Noteworthy in Release 11.3
  - New and Noteworthy in Release 11.2
  - New and Noteworthy in Release 11.1
  - New and Noteworthy in Release 11.0
  - New and Noteworthy in Release 10.2C
  - New and Noteworthy in Release 10.2B
  - New and Noteworthy in Release 10.2A
  - Eclipse platform upgrade
  - Class Browser
  - Eclipse terminology
  - Preferences and properties
  - Database connection profiles
  - The ABL Virtual Machine (AVM)
  - How to get started with Progress Developer Studio for OpenEdge
  - Getting help

**Tasks**

- Starting Progress Developer Studio for OpenEdge
- Setting workspace preferences
- Working with OpenEdge perspectives

**Reference**

- OpenEdge workspace preferences
- OpenEdge database connection wizard
- OpenEdge perspectives and views

**Chapter 2: Introducing OpenEdge Projects**

- Concepts
- Introducing OpenEdge Projects
- OpenEdge project and resource data files
Chapter 3: AppServer support in Progress Developer Studio for OpenEdge

Contents

Configuration variables ................................................................. 84
Project type options for application development ............................... 85
Project types in Progress Developer Studio for OpenEdge .................... 86

Tasks................................................................................................. 90
Creating a new OpenEdge project ...................................................... 90
Adding resources to a project ......................................................... 95
Adding existing database connections to a project ......................... 97
Adding a new database connection to a project ................................. 97
Setting project properties .............................................................. 98
Saving project property profiles ..................................................... 99
Changing project property profiles ................................................. 99
Defining new configuration variables ............................................ 104

Reference.......................................................................................... 105
New OpenEdge Project wizard ....................................................... 105
Progress OpenEdge project properties pages .................................... 111
Project Facets project properties page ............................................. 123
Progress OpenEdge Import and Export wizards ............................... 124
Build and Compile options ............................................................ 130

Chapter 3: AppServer support in Progress Developer Studio for
OpenEdge .......................................................................................... 135

Concepts.......................................................................................... 136
AppServer support in Progress Developer Studio for OpenEdge ........... 136
The AppServer in the OpenEdge Server perspective .......................... 137
AppServer launch configuration ...................................................... 138
Publishing ....................................................................................... 138
Project support for AppServers ....................................................... 139
Terminology related to AppServer support ....................................... 140

Tasks................................................................................................. 141
Accessing the OpenEdge Server perspective ..................................... 141
Setting Progress OpenEdge Server preferences ............................... 141
Defining an AppServer server ......................................................... 142
Adding an OpenEdge Explorer connection ..................................... 144
Defining the OpenEdge AVM runtime ............................................. 145
Viewing or modifying AppServer properties .................................... 146
Accessing OpenEdge Explorer ....................................................... 147
Working with AppServer connections and agents ........................... 147
Viewing AppServer status ............................................................. 149
Working with AppServer projects ................................................... 150
Publishing AppServer code to a remote server ................................. 153
Publishing AppServer code to a server for testing ............................ 154
Using the Clean option before republishing .................................... 154
Viewing a server’s log files ............................................................. 155

Reference.......................................................................................... 155
Chapter 6: Overview of ABLUnit testing framework ........................................... 243

- Concepts ........................................................................................................... 244
  - Overview of ABLUnit testing framework .................................................. 244
  - Test Class ..................................................................................................... 244
  - Test Suite Class .......................................................................................... 245
  - Test Procedure ............................................................................................. 246
  - Test Suite Procedure .................................................................................. 246
  - Annotations supported with ABLUnit ....................................................... 246
  - Lifecycle of ABLUnit framework ............................................................... 247
  - ABLUnit project type .................................................................................. 248

- Tasks ............................................................................................................... 248
  - Accessing ABLUnit perspective ............................................................... 248
  - Creating an ABLUnit project ................................................................. 248
  - Creating test cases using ABLUnit ...................................................... 249
  - Running test cases and test suites ....................................................... 249
  - Using the ABLUnit view ................................................................. 257

- Reference ...................................................................................................... 257
  - Test case generation wizards ............................................................ 257
  - ABLUnit view .......................................................................................... 261
  - Progress OpenEdge ABLUnit launch configuration settings ......... 261
  - Progress OpenEdge ABLUnit debug configuration settings .......... 262
  - Add New Test Method wizard ............................................................ 262
  - Add New Test Procedure wizard ....................................................... 263
  - Assert API .................................................................................................. 263

Chapter 7: Data Object overview .................................................................... 265

- Concepts ........................................................................................................ 266
  - Data Object overview ............................................................................... 266
  - Prerequisites for Data Object project support ........................................ 266
  - Terminology related to Data Object support ......................................... 267
  - OpenEdge Data Object project type .................................................... 269
  - Data Object project folder structure .................................................. 270
  - Data Object service artifacts ............................................................... 271
  - Data Object application WAR file structure .................................... 271

- Tasks .............................................................................................................. 273
  - Setting default preferences for starting servers automatically .......... 273
  - Working with Data Objects ................................................................. 274
  - Deploying Data Object Services .......................................................... 284
Chapter 15: Introducing the Meta Catalog

Concepts

ABL Annotations
Access to catalog data
Other OpenEdge Tools
Views
GUI Designer dialogs
Toolbar and menu options
Object Palette
Preferences page
AppBuilder perspective components
Editing a Freeform Query
Specifying DB-Required code blocks
Enabling AppBuilder shortcut keys
Opening AppBuilder Properties view
Working with method library reference
Running ABL procedure files
About custom object file
About Progress Dynamics
About Method Library
Common capabilities of SmartObjects
About SmartObjects
About custom object file
About Method Library
SmartObject instance
SmartObject master
Paging in SmartObjects
Layout
Tasks

Reference

AppBuilder perspective components
Preferences page
Object Palette
Toolbar and menu options
GUI Designer dialogs
Views
Other OpenEdge Tools
Chapter 16: Introducing Tools for Business Logic ............................ 851

Concepts............................................................................................ 852
Application modeling........................................................................ 852
Design principles and the OpenEdge Reference Architecture.............. 853
The Business Logic tool set................................................................. 853
Component types............................................................................... 853
Design output.................................................................................... 855

Tasks................................................................................................. 857
Setting Tools for Business Logic preferences................................. 857
Model database management......................................................... 858
Modeling components...................................................................... 861
Matching diagrams to ABL code..................................................... 868

Reference......................................................................................... 870
Component Designer......................................................................... 870
Application Schema Explorer.......................................................... 871
Outline view for diagrams............................................................... 872
Code to Model Component wizard................................................ 872
## Chapter 17: Introducing Progress Developer Studio for OpenEdge

### customization

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concepts</td>
<td>884</td>
</tr>
<tr>
<td>Menu and toolbar customization</td>
<td>884</td>
</tr>
<tr>
<td>Template customization</td>
<td>884</td>
</tr>
<tr>
<td>Startup customization</td>
<td>885</td>
</tr>
<tr>
<td>Event subscription</td>
<td>885</td>
</tr>
<tr>
<td>Custom project types</td>
<td>885</td>
</tr>
<tr>
<td>Tasks</td>
<td>886</td>
</tr>
<tr>
<td>Setting customization preferences</td>
<td>886</td>
</tr>
<tr>
<td>Using the Customization Editor</td>
<td>886</td>
</tr>
<tr>
<td>Customizing project startup</td>
<td>892</td>
</tr>
<tr>
<td>Reference</td>
<td>892</td>
</tr>
<tr>
<td>Subscribing to OpenEdge Architect events</td>
<td>892</td>
</tr>
<tr>
<td>Customization Editor</td>
<td>896</td>
</tr>
<tr>
<td>Customization Preferences page</td>
<td>904</td>
</tr>
</tbody>
</table>

## Chapter 18: Introducing Progress DB Navigator

### Concepts

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB Navigator perspective</td>
<td>908</td>
</tr>
<tr>
<td>SQL connection profiles</td>
<td>909</td>
</tr>
<tr>
<td>JDBC drivers</td>
<td>909</td>
</tr>
<tr>
<td>Multitenancy (Progress OpenEdge databases only)</td>
<td>909</td>
</tr>
<tr>
<td>Table partitioning in Progress OpenEdge databases</td>
<td>910</td>
</tr>
<tr>
<td>Tasks</td>
<td>910</td>
</tr>
<tr>
<td>Common tasks for supported databases</td>
<td>910</td>
</tr>
<tr>
<td>Working with MS SQL Server databases</td>
<td>920</td>
</tr>
<tr>
<td>Working with OpenEdge databases</td>
<td>924</td>
</tr>
<tr>
<td>Working with Oracle databases</td>
<td>951</td>
</tr>
<tr>
<td>Reference</td>
<td>957</td>
</tr>
<tr>
<td>Components of the DB Navigator perspective</td>
<td>957</td>
</tr>
<tr>
<td>MS SQL Server database</td>
<td>963</td>
</tr>
<tr>
<td>OpenEdge database</td>
<td>965</td>
</tr>
<tr>
<td>Oracle database</td>
<td>978</td>
</tr>
<tr>
<td>Preferences</td>
<td>982</td>
</tr>
</tbody>
</table>
Preface

For details, see the following topics:

- About this manual
- Typographical conventions

About this manual

This manual provides the Progress® Developer Studio for OpenEdge® online help as a PDF file. This PDF file contains the same information as the Progress Developers Studio for OpenEdge online help. However, the online help is preferable with regard to navigation and formatting. If you are running Progress Developers Studio for OpenEdge, you can access the online help from the Help menu.

Typographical conventions

This documentation uses the following typographical and syntax conventions:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bold</strong></td>
<td>Bold typeface indicates commands or characters the user types, provides emphasis, or the names of user interface elements.</td>
</tr>
<tr>
<td><em>Italic</em></td>
<td>Italic typeface indicates the title of a document, or signifies new terms.</td>
</tr>
<tr>
<td>SMALL, BOLD CAPITAL LETTERS</td>
<td>Small, bold capital letters indicate OpenEdge key functions and generic keyboard keys; for example, GET and CTRL.</td>
</tr>
<tr>
<td>Convention</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>KEY1+KEY2</td>
<td>A plus sign between key names indicates a <strong>simultaneous</strong> key sequence: you press and hold down the first key while pressing the second key. For example, <strong>CTRL+X</strong>.</td>
</tr>
<tr>
<td>KEY1 KEY2</td>
<td>A space between key names indicates a <strong>sequential</strong> key sequence: you press and release the first key, then press another key. For example, <strong>ESCAPE H</strong>.</td>
</tr>
</tbody>
</table>

**Syntax:**

<table>
<thead>
<tr>
<th>Fixed width</th>
<th>A fixed-width font is used in syntax, code examples, system output, and file names.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed-width italics</td>
<td>Fixed-width italics indicate variables in syntax.</td>
</tr>
<tr>
<td>Fixed-width bold</td>
<td>Fixed-width bold italic indicates variables in syntax with special emphasis.</td>
</tr>
<tr>
<td>UPPERCASE fixed width</td>
<td>ABL keywords in syntax and code examples are almost always shown in upper case. Although shown in uppercase, you can type ABL keywords in either uppercase or lowercase in a procedure or class.</td>
</tr>
<tr>
<td>Period (.) or colon (:))</td>
<td>All statements except <strong>DO</strong>, <strong>FOR</strong>, <strong>FUNCTION</strong>, <strong>PROCEDURE</strong>, and <strong>REPEAT</strong> end with a period. <strong>DO</strong>, <strong>FOR</strong>, <strong>FUNCTION</strong>, <strong>PROCEDURE</strong>, and <strong>REPEAT</strong> statements can end with either a period or a colon.</td>
</tr>
<tr>
<td>[ ]</td>
<td>Large brackets indicate the items within them are optional.</td>
</tr>
<tr>
<td>[ ]</td>
<td>Small brackets are part of ABL.</td>
</tr>
<tr>
<td>{ }</td>
<td>Large braces indicate the items within them are required. They are used to simplify complex syntax diagrams.</td>
</tr>
<tr>
<td>{ }</td>
<td>Small braces are part of ABL. For example, a called external procedure must use braces when referencing arguments passed by a calling procedure.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>Ellipses indicate repetition: you can choose one or more of the preceding items.</td>
</tr>
</tbody>
</table>
Introducing Progress Developer Studio for OpenEdge

Progress Developer Studio for OpenEdge® is an integrated ABL development environment that helps you to:

• Quickly build user-interface and business logic
• Incorporate and manage application data sources
• Run, debug, and deploy code

Progress Developer Studio for OpenEdge consists of a set of editors, views, dialogs, and wizards that run in the Eclipse framework. Because Eclipse is an open-source, plug-in based framework, you can integrate a wide variety of software tools into the Eclipse installation that supports Progress Developer Studio for OpenEdge. See the Eclipse Plugin Central Web site for more information.

For details, see the following topics:

• Concepts
• Tasks
• Reference
Concepts

Introducing Progress Developer Studio for OpenEdge

Progress Developer Studio for OpenEdge® is an integrated ABL development environment that helps you to:

• Quickly build user-interface and business logic
• Incorporate and manage application data sources
• Run, debug, and deploy code

Progress Developer Studio for OpenEdge consists of a set of editors, views, dialogs, and wizards that run in the Eclipse framework. Because Eclipse is an open-source, plug-in based framework, you can integrate a wide variety of software tools into the Eclipse installation that supports Progress Developer Studio for OpenEdge. See the Eclipse Plugin Central Website for more information.

What’s New in Release 11.7

Release 11.7 contains Progress Developer Studio for OpenEdge updates to provide the following features:

• **New OpenEdge Project wizard**—The interface for the OpenEdge Project wizard has changed. The Create an OpenEdge Project page on page 105 now displays Server, Client, and General categories for project type selection. Do one of the following:
  
  • To create a project for the back-end development of the application, select the Server category from the left section of the page. Select the server type that you want to deploy the services to and the transport type that you want to use for deploying the application.
  
  • To create a project for the front-end development of the application, select the Client category from the left section of the page. Select the Desktop Application or the Web Application option button to create a project for a desktop application or a web application respectively and select one of the project types.
  
  • To create a basic, ABLUnit, Dynamics, or custom OpenEdge project, select the General category from the left section of the page.

  **Note:** If you do not select a category (Client, Server, or General) and click Finish, a basic OpenEdge project without any specialization is created.

• **Debugger support**—You can now use the Debug Configurations wizard to debug ABLUnit and Progress Application Server for OpenEdge.

• **Log Viewer**—The Log Viewer helps you to view and monitor the log files of a PAS for OpenEdge server instance from the Servers view. There are multiple log files for a server, you can choose to go to the directory where the log files are saved and manually search for errors, or view from Developer Studio using the Log Viewer.

• **ANT tasks for Progress Application Server for OpenEdge**—You can write custom ANT tasks to package ABL Web App and Web UI projects and to generate a Data Object Service Catalog file.

• **Migration of Spring Security files**—If you imported an ABL Web App project from OpenEdge 11.6 to the current release in the Developer Studio, you can also migrate the Spring Security files.
• **Infragistics control upgrade**—The Infragistics controls are upgraded to version 16.2.20162.2040.

• **Telerik UI for WinForms control upgrade**—The Telerik UI for WinForms controls that installs by default is upgraded to R3 2016 SP1 (2016.3.1024). If you need a different version, you can download it from the Telerik website.

• **Eclipse version update**—Eclipse version has been updated to 4.5.2.

---

**New and Noteworthy in Release 11.6**

Release 11.6 contains Progress Developer Studio for OpenEdge updates to provide the following features:

• **WebSpeed support in PAS for OpenEdge**—PAS for OpenEdge is available as a host for WebSpeed applications. You can migrate existing WebSpeed applications to PAS for OpenEdge, or you can use the Progress Developer Studio for OpenEdge to develop new WebSpeed applications that run on PAS for OpenEdge.

  You can create an ABL Web App to deploy one or more ABL services (of WebSpeed, REST, or Data Object types) as a single web app to Progress Application Server for OpenEdge.

  The `web-disp.p` control program available for WebSpeed applications for traditional servers does not exist in PAS for OpenEdge. Web object execution is controlled by a built-in handler object. This default handler class can be modified to implement any `web-disp.p` customizations that you want to replicate. However, you cannot migrate `web-disp.p` to a PAS for OpenEdge instance.

• **OpenEdge Data Object changes**—Mobile services is now rebranded as OpenEdge Data Object Services.

  Progress Developer Studio for OpenEdge provides wizards and other tools to accelerate the development of server-side Data Objects and Data Object Services.

  Progress Developer Studio for OpenEdge allows you to develop and deploy Data Object Services that you can use to provide OpenEdge data to mobile apps running in native device environments or to web apps, which can run in any browser-based environment.

  OpenEdge Mobile Builder is no longer supported. You can use Telerik Platform to create mobile apps.

  You can develop Data Objects Services using:

  • Data Object projects that support Data Object Service development using facets that work with the OE AppServer and a Tomcat web server that hosts the Data Object Services. For more information on working with the Data Object project and creating Data Object Services, see the *Data Object* help.

  • ABL Web App projects that support Data Object Service development using facets that work with PAS for OpenEdge only. For more information on working with the ABL Web App project and creating Data Object Services, see the *Progress Application Server for OpenEdge* help.

  **Note:** For more information on using OpenEdge Data Objects to access OpenEdge data from mobile and https://documentation.progress.com/output/pdo location.

• **Profiler Editor**—Progress Developer Studio now provides an built-in profiler editor. You can use the Profiler tab in Run Configuration to profile a session. You can view the details of the profiled session in the editor.

• **References Update to Telerik assemblies**—You can update references to Telerik assemblies if you have upgraded to a new version of Progress Developer Studio for OpenEdge, and you want to update all references to work with the new version.
Progress Developer Studio for OpenEdge updates the assemblies automatically the first time you open a workspace. However, if you decline to perform the automatic update, later when you launch the Update Assembly References tool from the Progress Developer Studio for OpenEdge menu, all instances of assemblies.xml and toolbox.xml in your current workspace automatically appear in the list of files to be updated, and all of the necessary settings are in place by default.

- **Telerik UI for WinForms**—Progress Developer Studio for OpenEdge installs a trial version of Telerik UI for WinForms by default. It includes controls that you can use to build your business applications. If you need a different version of Telerik, you can download it from the Telerik web site.

  The Toolbox in the Visual Designer Editor now includes all the Telerik UI for WinForms controls.

- **Assert API in ABLUnit**—The Assert API provides a set of assertion methods to validate the expected result of a method or procedure in a test with that of the actual result. This API is a part of the OpenEdge.Core.Assert, OpenEdge.Core.Assertion.AssertFile, OpenEdge.Core.Assertion.AssertJSON, OpenEdge.Core.Assertion.AssertError, and OpenEdge.Core.Assertion.AssertObject classes in the OpenEdge.Core.pl file. For detailed information on all classes and methods in Assert API, see https://documentation.progress.com/output/oehttpclient/index.html

- **Infragistics controls upgrade**—The Infragistics controls are upgraded to version 15.1.20151.2132.

### New and Noteworthy in Release 11.5

Release 11.5 contains Progress Developer Studio for OpenEdge updates to provide the following features:

**ABLDoc support**

With this release, Progress Developer Studio for OpenEdge supports generating ABLDoc documentation using the new Generate ABLDoc wizard or by using a custom ABLDoc Ant task.

You can generate ABLDoc documentation for the following source codes:

- Class (.cls) files
- Procedure (.p) files
- Include (.i) files

For more information, see the ABL Editor help.

**Progress Application Server integration**

Progress Application Server is a platform that provides Web server support for Progress applications. Progress applications are packaged as Web application archives (WAR files) and deployed to the Java Servlet Container of a running instance of PAS. Client access to a PAS is through HTTP/HTTPS protocols. Clients include (but are not limited to) browser-based applications and Mobile Apps.

Progress Application Server for OpenEdge is a Web application server based on the Apache Tomcat Web server and servlet container. Progress Developer Studio for OpenEdge now provides an environment to add Progress Application Server for OpenEdge instances to the Servers view. You can also control the state of a Progress Application Server for OpenEdge and deploy applications onto a Progress Application Server that is configured and selected in the Servers view.

For more information, see the Progress Application Server for OpenEdge help.
Support for creating partitioned tables
Progress Developer Studio for OpenEdge now lets you to create a partitioned table (with local indexes) without any partition policies defined on it. This support is only available to a connected OpenEdge database that is enabled for table partitioning. After creating the partitioned table, you can use OpenEdge Management, OpenEdge Explorer, or OpenEdge SQL to define partition policies on it.

For more information, see the DB Navigator help.

Support for exporting Mobile Apps as debug binary or release binary files
With this release, Progress Developer Studio for OpenEdge lets you set the Mobile App Builder properties to export a Mobile App as a release binary or a debug binary file. You can also set the properties to select the platform and specify the target in which you want to build the Mobile App.

For more information, see the Mobile help.

Support for new mapping definitions in REST Resource URI Editor
The Mapping Definitions section of the REST Resource URI Editor has been updated to include a new Content-Type header in the Input and Output tabs. The Output tab also includes a new Constants node, under the Advanced node, which can be used to add a constant (like application/myApp+json) and map it to the Content-Type header.

For more information, see the REST help.

Infragistics controls upgrade
In this release, the Infragistics controls have been upgraded to version 14.2.20142.2010.

See also
New and Noteworthy in Release 11.4 on page 21
New and Noteworthy in Release 11.3 on page 24
New and Noteworthy in Release 11.2 on page 25
New and Noteworthy in Release 11.1 on page 27
New and Noteworthy in Release 11.0 on page 28
New and Noteworthy in Release 10.2B on page 35

New and Noteworthy in Release 11.4
Progress Developer Studio for OpenEdge 11.4 has been updated to provide the following features

ABLUnit testing framework
ABLUnit testing is a testing methodology that involves writing small tests that execute a small portion of an application and checks whether it is working as intended. The ABLUnit testing framework is similar to an xUnit or a JUnit testing framework.

An ABLUnit testing framework enables you to:

• Identify bugs in the code
• Write modular and loosely-coupled code
• Make changes to the code
• Test the code in parts
Before-image data support

The Define Service Interface and New Business Entity wizards in Progress Developer Studio for OpenEdge have been updated with new options to add the writeDataSetBeforeImage attribute to the Mobile interface annotations and to expose a file as a Mobile resource.

Block level error handling

The New ABL Class wizard in Progress Developer Studio for OpenEdge now provides a Block level error handling option that adds the BLOCK-LEVEL ON ERROR UNDO, THROW statement to the generated ABL code.

Exclusion of PROPATH entries from OpenEdge tooling

Progress Developer Studio for OpenEdge introduces a new option, Exclude from OpenEdge tooling, which enables you to exclude PROPATH entries from OpenEdge tooling.

Express JSDO templates

Progress Developer Studio for OpenEdge now enables you to create an express Mobile project with JSDO services.

The following new templates have been added to the Create an Express Mobile application page for this purpose:

- Express JSDO Phone App
- Express JSDO Tablet App

Freeform query support

Progress Developer Studio for OpenEdge has extended the freeform query support by adding the following:

- A new decorator in the Outline view to distinguish a freeform query node from other nodes
- Support for editing a freeform query and updating the appropriate code sections in the source

Improved typing assistance in ABL Editor

ABL Editor now supports auto-completion for quotes and double quotes.

JSDO Catalog enhancement

The JSDO catalog has been extended to support additional properties to expose OpenEdge services as Mobile services.

Push notification service

Progress Developer Studio for OpenEdge now enables you to use ABL API to send an immediate notification to a registered iOS and/or Android device, and schedule a notification for a later time. To be able to send and receive the notifications, you must enable the push notification service for the Mobile App and provide the API key and ID in the Mobile App Builder, and deploy the App as a Native App on the registered device.

Support for associating a project with an existing Mobile App

With the improved Mobile App Builder integration with Progress Developer Studio for OpenEdge, you can now associate an existing Mobile App in the Mobile App Builder with a project in the Progress Developer Studio for OpenEdge workspace.
To enable this feature, a new option, **Associate with an existing Mobile App**, has been introduced in the **New Mobile App** wizard. You can now use the **New Mobile App** wizard to either create a new Mobile App using a template or a backup file, or associate an existing one with the selected project. This wizard also enables you to create a session-enabled Phone or Tablet App.

**Support for finding references of ABL constructs**

With this release, Progress Developer Studio for OpenEdge introduces the capability of finding all references of specific ABL constructs (like variables, methods, etc.) in an ABL file, project, or workspace.

**Support for marking a class as serializable**

The **New ABL Class** wizard now provides a new **Serializable** option that enables you to mark a class as serializable by including the `SERIALIZABLE` modifier in the generated ABL code.

**Support for organizing USING statements**

Progress Developer Studio for OpenEdge now enables you to organize the USING statements in ABL sources through a set of default rules or custom rules that you can define by using a set of preferences. These rules include cleaning, sorting, and grouping USING statements.

**Table partitioning support in DB Navigator**

The **DB Details** view of DB Navigator has been updated to indicate whether a table is partitioned or not and if a database contains partitioned tables. The **DB Details** view now displays icons to distinguish between a partitioned table with policies defined on it and a partitioned table without policies defined on it. It also displays icons for the local index of a partitioned table and the partition key based on which the table is partitioned.

**Upgrade to Eclipse 4**

Progress Developer Studio for OpenEdge now supports Eclipse 4. Eclipse 4 is the next generation platform for building Eclipse-based tools and rich client desktop applications. This new release makes it easier to develop and assemble applications and tools based on the Eclipse platform.

Eclipse 4 introduces the following features:

- A model-based user interface and a new CSS-based declarative mechanism for application styling. These facilities make it easier to design and customize an application user interface. This also adds greater flexibility to UI layout, making it easier to make a user interface look different from an IDE.
- A new services-oriented programming model that makes it easier to use discreet application services of the Eclipse platform.
- A compatibility layer to allow existing well-behaved Eclipse 3.x applications to take advantage of the new functionality of the Eclipse 4 Application Platform.

**See also**

- [New and Noteworthy in Release 11.3](#) on page 24
- [New and Noteworthy in Release 11.2](#) on page 25
- [New and Noteworthy in Release 11.1](#) on page 27
- [New and Noteworthy in Release 11.0](#) on page 28
- [New and Noteworthy in Release 10.2B](#) on page 35
New and Noteworthy in Release 11.3

The 11.3 release of Progress Developer Studio for OpenEdge includes new and revised features in the following areas:

**Express Mobile project**
Progress Developer Studio for OpenEdge now allows you to automatically build and deploy a complete Mobile application for a given database table and Mobile App template using the **Express setup** option.

**OpenEdge Business Rules**
The OpenEdge and Corticon integration enables you to use ABL data structures (such as ProDataSets and temp-tables) as Corticon data structures (Vocabularies and entities). This integration supports OpenEdge Business Rules that provide tooling support for creating and updating Corticon Vocabularies, and runtime support for simple invocation of Corticon Decision Services.

**Database table selection in the Business Entity wizard**
Progress Developer Studio for OpenEdge now provides support for generating schema and implementing default CRUD operations dynamically while creating a Business Entity. In addition to the current schema definition selection from the schema file, you can now select a database table from a connected database and a set of fields from the database. The selected table and corresponding table fields are used for the temp-table definition dynamically and copied to the Business Entity class.

**Infragistics controls update**
The Infragistics controls have been upgraded to version 13.1.20131.2015.

**Infragistics controls help support**
You can now access the online help for Microsoft and Infragistics controls, which is available on their respective sites, by pressing F1.

**Win 64 support**
Progress Developer Studio for OpenEdge is now available on the Windows 64-bit platform.

**Support for special contexts in mapper**
With this release, Progress Developer Studio for OpenEdge has added the following context parameters to the **Input** tab of the **REST Expose Editor**.

- HTTP Headers
- HTTP Servlet Request
- HTTP Servlet Response
- Servlet config and Servlet context

These context parameters are grouped under the **Server Contexts** node.

**Incremental PAAR support**
Progress Developer Studio for OpenEdge now allows you to export services incrementally by creating a .PAAR file for each REST or Mobile service in the project.
Seamless integration of OEBPM plugins in Progress Developer Studio for OpenEdge

You can now use a unified installer for Progress Developer Studio for OpenEdge and OEBPM.

See also

New and Noteworthy in Release 11.2 on page 25
New and Noteworthy in Release 11.1 on page 27
New and Noteworthy in Release 11.0 on page 28
New and Noteworthy in Release 10.2B on page 35

New and Noteworthy in Release 11.2

The 11.2 release of Progress Developer Studio for OpenEdge includes new and revised features in the following areas:

- OpenEdge REST application support
- OpenEdge Mobile application support
- OpenEdge AppBuilder integration
- ABL Scratchpad view

OpenEdge REST application support

OpenEdge 11.2 provides the ability for REST clients to initiate the execution of business logic on an OpenEdge AppServer, and to receive back results from that execution. REST (Representational State Transfer) is an architectural style for distributed computing that was defined in conjunction with the HTTP protocol. A REST client is similar to any other OpenEdge client, and uses HTTP as its communication protocol.

REST support in OpenEdge includes:

- Application development tools in Progress Developer Studio for OpenEdge (Developer Studio)
- Testing on a built-in Web server (Apache Tomcat) in Developer Studio
- Configuration and management through command line tools or OpenEdge Explorer/OpenEdge Management
- Secure REST application configuration supported using the Spring Security framework
- Publishing REST applications to external Web servers

OpenEdge Mobile application support

OpenEdge Mobile allows you to build a complete mobile application, including the mobile UI, with access to AppServer business logic and data that you can deploy on a variety of popular mobile and Web devices and platforms, including:

- Apple iOS devices, such as the iPhone and iPad
- Android devices, including numerous smart phones and tablets made by a variety of manufacturers
- Web browsers running on a wide variety of computers and mobile devices
- Apache Tomcat Web server for REST access to the AppServer

OpenEdge Mobile supports development for two basic types of mobile UI (Mobile App):

- Web apps that run in a Web browser on multiple platforms and devices
- Native apps that are built the same as Web apps, but that are deployed to run in native device containers
You can build and test all application components end-to-end, including the AppServer and corresponding Mobile services, the Mobile App, and the Web transport between them, using OpenEdge Mobile tools available through Progress Developer Studio for OpenEdge (PDSOE). When your application development is complete, you can then deploy the application components as required to supported mobile devices and Web platforms.

**OpenEdge AppBuilder integration**

The 11.2 Release of Progress Developer Studio for OpenEdge (Developer Studio) includes the integration of the OpenEdge AppBuilder. The new improved seamless AppBuilder integration makes Developer Studio a one-stop solution for developing and maintaining ABL GUI applications.

The AppBuilder integration introduces a new design editor for designing AppBuilder GUI applications called ABL UI Designer. This design editor allows developers to draw widgets and generate source code, similar to the Visual Designer that is used to build GUI for .NET applications.

With this integration, the following OpenEdge AppBuilder capabilities can be accessed from within the Developer Studio:

- OpenEdge AppBuilder perspective to build AppBuilder GUI applications within an OpenEdge project.
- AppBuilder options available as AppBuilder preferences and project properties in the Developer Studio. These options allow developers to control the behavior of AppBuilder and ABL UI Designer such as Dynamics, Editor, Grid Units, and Widget ID settings.
- ABL GUI applications are OpenEdge project-aware. That is, developer can set common properties for all the ABL GUI applications defined in an OpenEdge project using AppBuilder properties.
- AppBuilder templates and wizards to develop new ABL GUI applications. Stand-alone AppBuilder templates and wizards run embedded in Eclipse standard wizards from within the Developer Studio.
- Design ABL GUI applications using the ABL UI Designer editor which extends the native AppBuilder functionality such as dock-able widget palette, look and feel of Visual Designer editor, hierarchical widget Outline view, and synchronization between the ABL UI Designer and ABL Editor.
- Improved ABL Editor to make sections of the AppBuilder-generated code read-only and foldable. This ensures that developers do not inadvertently modify the AppBuilder-generated code while editing the ABL application code is the ABL Editor.
- AppBuilder Section Editor to view specific block of code using the object nodes created in the Outline view. You can use the Section Editor Toggle button to enable or disable the Section Editor capability.
- Progress Dynamics features such as Open Dynamics Object, Register to Repository, Save Dynamics Object as Static, Save Static Object as Dynamic, and Open Associate Procedure.
- PRO*Tools utilities categorized into these three menu options: General PRO*Tools, AppBuilder PRO*Tools, and ADM2 PRO*Tools.

The OpenEdge AppBuilder can either run embedded in Progress Developer Studio for OpenEdge or as a standalone application.

**ABL Scratchpad view**

The 11.2 release of Progress Developer Studio for OpenEdge includes a new multi-tabbed view called ABL Scratchpad that allows you to write and test the ABL code without having to save it or creating a new file or project. This saves time and avoids the need to create a project or document multiple times for testing the code in different runtimes.

The ABL Scratchpad view appears in all the OpenEdge perspectives except the DB Navigator perspective. By default, this view appears at the bottom of the page in all the perspectives.

**See also**

New and Noteworthy in Release 11.1 on page 27
New and Noteworthy in Release 11.1

The 11.1 release of Progress Developer Studio for OpenEdge includes new and revised features in the following areas:

- Eclipse version update
- New parameters added to integrateArchitect.bat
- Opening multiple files in the same instance of Progress Developer Studio for OpenEdge
- Debug Listing Support
- Removal of Classic Software Updates
- Optimally build the Abstract Syntax Tree (AST) for an Include file
- Upgrade to Infragistics .NET v2012
- DB Navigator

Eclipse version update

The 11.1 release of Progress Developer Studio for OpenEdge is shipped with Eclipse 3.6 and is certified with Eclipse 3.7.1. Both these versions are supported by Progress Software Technical Support.

New parameters added to integrateArchitect.bat

With the 11.1 release, the following new parameters have been added to the integrateArchitect.bat file:

- OVERWRITE_EXISTING_PARAMS: If set to TRUE, the file property (provided by Progress Developer Studio for OpenEdge) set in eclipse_orig.ini will be added with the value specified in the eclipse.ini file (provided by Progress Developer Studio for OpenEdge). If the property is not set, it will not result in any change in the target eclipse.ini file.
- ADD_MISSING_PARAMS: If set to TRUE, the current property value (provided by Progress Developer Studio for OpenEdge) in the target Eclipse is overwritten with the newly provided value.

These parameters are turned off by default. The eclipse_orig.ini and config_orig.ini files are located in the %DLC%/oeide location. The integrateArchitect.bat uses these files and merges them into the destination Eclipse files.

Opening multiple files in the same instance of Progress Developer Studio for OpenEdge

With the 11.1 release, you can open multiple files in the same instance of Progress Developer Studio for OpenEdge. The files that support this feature are: .p, .i, .w, and .dgm (diagram files). For opening other file extensions, you must have a registry entry.

To open files in the same instance of Progress Developer Studio for OpenEdge, you can use one of the following methods:

- Select the files, right-click and select Open in PDSOE
- Select the files and double-click
- Use the command line
Debug Listing Support
The 11.1 release of Progress Developer Studio for OpenEdge allows you to use the line number of a debug listing file to go directly to a specific source line of the currently active file in the OpenEdge ABL Editor. You no longer need to manually generate the debug listing file in order to identify the actual location of the error in the source code.

Removal of Classic Software Updates
From the 11.1 release of Progress Developer Studio for OpenEdge, the Classic Software Updates option is removed from the Progress Developer Studio for OpenEdge Help menu. Also, the Classic Software update preferences option is removed from Windows > Preferences > Progress OpenEdge > Advanced, which was used to enable Classic Software Updates for the OpenEdge application installed on your system.

Instead of the Classic Software Updates menu option, you can use the Install New Software menu option or P2 director application to install a new plugin or software upgrade for Progress Developer Studio for OpenEdge.

Optimally build the Abstract Syntax Tree (AST) for an Include file
Some of the programs might have many references of the same include file. Such programs cause the Progress Developer Studio for OpenEdge to consume more memory, because each include file has a separate AST. The Include file preferences section on the Parsing preferences page allows you manage the include file references of a program while constructing an AST.

You can access the Parsing preferences page by selecting Window > Preferences > Progress OpenEdge > Advanced > Parsing.

Upgrade to Infragistics .NET v2012
The 11.1 release of Progress Developer Studio for OpenEdge includes Infragistics .NET v2012 Vol2. The 11.0 release was shipped with Infragistics .NET v2011 Vol1.

DB Navigator
DB navigator, in the 11.1 release, has been enhanced as described in the following sections.

Support for defining area for default tenant in DB Navigator
DB Navigator has been enhanced to support associating a multi-tenant table with a storage area. A new option, Keep area for default tenant, has been added to the Add Table and Edit Table wizards.

DB Navigator upgrade to JDBC 5.0 driver
To use the enhanced functionality and to remain current with the upgraded drivers, DB Navigator has been upgraded to the JDBC 5.0 driver.

See also
New and Noteworthy in Release 11.0 on page 28
What's New in Release 10.2B on page 35
Eclipse platform upgrade on page 43
Class Browser on page 48

New and Noteworthy in Release 11.0
New and Noteworthy in Release 11.0
The 11.0 release of Progress Developer Studio for OpenEdge (OpenEdge Architect) includes new and revised features in the following areas:

- **OpenEdge Architect renamed as Progress Developer Studio for OpenEdge**
- **Upgrade to Eclipse 3.6.0**
- **Debugging**
- **WebSpeed**
- **AppServer**
- **Visual Designer**
- **Support for ABL SIGNATURE-ONLY events**
- **Enhancements to ProBindingSource Designer**
- **DB Navigator**
- **Custom project types**
- **Excluding files and folders from compilation**

**OpenEdge Architect renamed as Progress Developer Studio for OpenEdge**

Beginning with the 11.0 Release of OpenEdge, Progress Developer Studio for OpenEdge replaces OpenEdge Architect as a product name. Progress Developer Studio for OpenEdge is not a new product. References in documentation to Progress Developer Studio for OpenEdge, or OpenEdge Architect refer to the same product.

Also be aware that you no longer choose Start > OpenEdge > OpenEdge Architect to launch the product from the Windows Start menu. In the 11.0 release you can start Progress Developer Studio for OpenEdge by selecting one of the following default locations:

- **Start > Progress > OpenEdge 11 > Progress Developer Studio 3.6 for OpenEdge**
- **Start > Progress > OpenEdge 11 > Progress Developer Studio 3.6 for OpenEdge Clean**
- **Start > Progress > Developer Studio 3.6 > Progress Developer Studio 3.6**
- **Start > Progress > Developer Studio 3.6 > Progress Developer Studio 3.6 Clean**

**Note:** Clean is a startup option that removes cached data before starting Eclipse.

**Upgrade to Eclipse 3.6.0**

The 11.0 Release of Progress Developer Studio for OpenEdge includes Eclipse 3.6.0 (The 10.2B release was shipped with Eclipse 3.4.2.)

**Debugging**

The following are the new debug launch configuration features provided in the 11.0 Release of Progress Developer Studio for OpenEdge:

- **AppServer and WebSpeed debugging** - Remote and local debugging of both AppServer and WebSpeed applications.
- **Attachable Debugger** - A new debug launch configuration type that allows you to attach the debugger to an external AVM.
- **AppServer pass-through** - Debugging a remote procedure from within Architect, during pass-through, without having to set breakpoints in the remote procedure code.
• Standalone Debugger functionality - The debugging features provided by the standalone debugger are now available within Progress Developer Studio for OpenEdge.

With the enhanced debugging functionality, you can:

• Debug code running on a local or remote AVM from within Progress Developer Studio.
• Set breakpoints in ABL files located outside the workspace, and also in include files.
• Debug multiple AppServer or WebSpeed agents in a single Progress Developer Studio session.
• Attach the Debugger to an AppServer or WebSpeed broker in debug mode.
• View the ABL stack for multiple AVMs in the Debug view.
• Debug SpeedScript applications running on a WebSpeed agent.
• Launch WebSpeed brokers with the new run and debug launch configurations.

WebSpeed
The 11.0 Release of Progress Developer Studio for OpenEdge includes improved support for creating and managing WebSpeed applications. WebSpeed support includes:

• The addition of a WebSpeed project type to the OpenEdge project wizard.
• Templates and wizards for creating SpeedScript and CGI Wrapper files.
• Full-featured editors for editing HTML, Embedded SpeedScript, and CGI Wrapper files.
• Content assist for super procedures.
• WebSpeed server support for running, debugging, and testing WebSpeed applications.
• Support for the configuring of, and publishing to local and remote servers. (You can optionally compile to the AVM of a remote broker when you publish.)

AppServer
Enhancements to AppServer support in the 11.0 Release of Progress Developer Studio for OpenEdge allow you to:

• Create and publish an application to a local or remote AppServer, so you can test the application in an environment outside of the Progress Developer Studio workspace.
• Use OpenEdge Explorer, which provides an HTTPS connection for secure communication, to create and manage AppServer brokers.

You can also use OpenEdge Explorer to publish to a remote machine (minimum version OpenEdge 11.0 required). You can access OpenEdge Explorer from within OpenEdge Developer Studio.

• Use OpenEdge Explorer to publish to a remote machine (minimum version OpenEdge 11.0 required). You can access OpenEdge Explorer from within Progress Developer Studio for OpenEdge.
• Optionally compile as you publish, when publishing remotely.
• Create custom project types.
• Monitor all brokers from a single view.
• Associate a project with both AppServer and WebSpeed if the project contains the required facet.

Visual Designer
Visual Designer, in the 11.0 Release of Progress Developer Studio for OpenEdge, is updated to provide support for ABL SIGNATURE-ONLY events and is enhanced with an improved ProBindingSource Designer.
Support for ABL SIGNATURE-ONLY events

In earlier releases of OpenEdge, the Visual Designer Properties view displayed the public .NET events including events defined in ABL using the DELEGATE clause. In OpenEdge 11.0, the Visual Designer Properties view displays the ABL events that are defined in an ABL-derived .NET class using the DELEGATE or the SIGNATURE clause. The events using the SIGNATURE clause appear in the Events tab only if their signatures follow the .NET convention for event handler signatures.

Enhancements to ProBindingSource Designer

In earlier releases of OpenEdge, the ProBindingSource Designer opened multiple modal dialogs to define a schema. The Available Schema panel is now available on the ProBindingSource Designer window. It enables you to define schema more easily and reduces the number of steps for importing schema.

In addition, support for defining recursive relationships between tables is added to the interface. A combo box labeled Recursive table lists the name of the selected table and all the parent tables where a recursive relationship can be defined.

ABL Editor

The ABL Editor, in the 11.0 Release of Progress Developer Studio for OpenEdge, is enhanced to enable or disable processing preprocessors from a toggle button on the toolbar. The following features are supported when preprocessing is enabled:

- The ABL Editor displays the inactive preprocessor regions in gray.
- Code assistance displays proposals for the built-in or defined preprocessors. Preprocessor references are expanded when evaluating completion proposals.
- Hovering over the preprocessor reference displays the current value of the preprocessor in a pop-up dialog.
- The elements inside the inactive preprocessor regions appear in gray in the Outline view.
- The Quick Outline view displays a list of preprocessors and their values.

In addition, the ABL Editor supports the following ABL language changes:

- Color-coding recognizes the DYNAMIC-PROPERTY keyword, and the SetPropertyValue() and GetPropertyValue() methods. The ABL Editor displays these syntax elements in different colors to make them easy to recognize.
- Content assistance displays a list of syntax elements that are valid for the DYNAMIC-PROPERTY keyword, and the SetPropertyValue() and GetPropertyValue() methods. You can select a syntax completion proposal from the list to complete the code in the ABL Editor.
- Hovering over the DYNAMIC-PROPERTY keyword displays additional information in a pop-up dialog.
- A new ABL Interface now inherits interfaces from other interfaces.

DB Navigator

DB Navigator was enhanced to support the multi-tenant features in OpenEdge databases. DB Navigator now enables:

- Serving multiple users with a single OpenEdge database instance.
- Creating, editing, and identifying multi-tenant tables and sequences.
- Categorizing and filtering DB tables.
- Launching OpenEdge Explorer from the DB navigator perspective for configuring the database.
- Sharing data definition files (schema changes) among developers.
Custom project types

OpenEdge projects and the Customization Editor were enhanced in OpenEdge Release 11.0 to support custom project types.

The following enhancements are now available for OpenEdge projects:

• Progress Developer Studio for OpenEdge creates all new OpenEdge projects as faceted projects and converts the old OpenEdge projects into faceted projects by default.

• Progress Developer Studio for OpenEdge provides a predefined set of OpenEdge facets and project types.

• Progress Developer Studio for OpenEdge allows you to create a specialized project type for the current project with the enhanced New OpenEdge Project wizard.

• Progress Developer Studio for OpenEdge provides a set of commonly used project types for OpenEdge application development and a framework to configure custom project types as required.

The enhancements made to the Customization Editor now allow you to:

• Create custom project types with the desired folder layouts and PROPATH settings.

• Generate new Editor templates.

• Configure code-generation templates and use them in custom projects.

• Share custom project types with other developers.

• Use custom project types as templates for new projects.

Excluding files and folders from compilation

In earlier releases of Progress Developer Studio for OpenEdge, there was no option to exclude a file or folder from a build. The build and compile environment in Progress Developer Studio for OpenEdge is enhanced as follows:

• You can exclude project resources from your build from the context menu (Progress OpenEdge > Exclude from Build) or from the Source tab of the Build properties page. Excluded entries are added to the Excluded node in the Source tab of the Build properties page.

• You can include the resources that were previously excluded from the build from the context menu (Progress OpenEdge > Include in Build) or from the Source tab of the Build properties page.

• You can create working sets to compile a set of resources instead of building the entire project.

• You can control the build order using the Source tab of the Build properties page.

See also

What's New in Release 10.2B on page 35
Eclipse platform upgrade on page 43
Class Browser on page 48

New and Noteworthy in Release 10.2C

This section describes the new features and changes that were implemented in the 10.1C release of OpenEdge Architect (renamed as Progress Developer Studio for OpenEdge in Release 11.0).

• Class Browser

• Progress DB Navigator
Class Browser
A new view called the Class Browser allows you to see the structure and content of the various classes that you work with in creating ABL applications. The Class Browser's contents are organized by resources, which are collections of classes (types).

Progress DB Navigator
In order to make it more independent, the OpenEdge DB Navigator plug-in has been slightly restructured and was renamed the Progress DB Navigator. You will notice a change in the location and structure of its Preferences as well as a renaming of the perspective. In addition to continuing to provide the driver for the OpenEdge database, Progress Software now provides Type IV DataDirect JDBC drivers for the following databases: ORACLE, Microsoft SQL Server, IBM DB2, Sybase, and Informix.

AB Editor

Support for structured error handling
To support ABL enhancements in error-handling functionality, OpenEdge wizards, menus, and context assistance now provide options for automatically generating:

- Routine-level error-handling statements in new classes and procedures
- CATCH and FINALLY blocks in new or existing source files

Performance-related options for code assistance
The Editor Assistance preferences page includes several new options that can mitigate adverse impact on system response time caused by waiting for the code to be analyzed:

- You can specify the maximum amount of time that OpenEdge spends updating code analysis before responding to requests for code-completion assistance. If the re-parsing of the code is not completed within the specified interval (by default 300 milliseconds), OpenEdge proposes completion options based on the most recent full analysis.

- You can disable highlighting of matching elements. That is, you can suppress the display of a marker at the position of the code element, such as a parenthesis or a beginning or ending statement in a code block, that is paired with the element at the current cursor position.

- You can suppress field descriptions in completion proposals for database tables.

Using these options can improve performance, particularly when you work with very large source files or remote databases.

Support for static data members, methods, and properties
To support the use of the STATIC keyword where applicable, OpenEdge macros, wizards, and menus now provide options for automatically defining data members, methods, and properties as static.

In the Outline view, the icons for static data members, methods, and properties appear with a red "S" overlaying the standard image.

Improved support for coding methods implemented via interface
The New Class wizard has been enhanced to automatically generate method stubs for all methods implemented via interface. In addition, the former Override Methods dialog, now called Override/Implement Methods, has been enhanced to generate stubs for interface methods on demand.
Enhanced macro functionality

OpenEdge now uses standard Eclipse template functionality for creating and managing macros (also called templates). Procedures for using and managing macros are largely unchanged from previous versions, but the enhanced design offers these advantages:

- You can now define and run multiple-line macros. Previously, macros were restricted to a single line.
- For creating custom macros, the New Template editor includes an Insert Variable command that improves usability.

To make custom macros created in previous versions of OpenEdge available for use in version 10.1C, migration is required. When you first select a workspace created in an earlier version, OpenEdge offers the option of migrating the macros for that workspace. If you decline to perform the migration at that time, you can do so at your convenience by selecting **OpenEdge>Migration>Migrate 10.1A or 10.1B Macros to Templates**.

Formatting changes

The following changes have been made to OpenEdge formatting and code-assistance options:

- The Format option no longer appears on the Source menu, its functions replaced by separate commands as described in the next two entries in this list.
- A new command (Source>Correct Case, Ctrl+Shift+F) causes the ABL Editor to adjust the case of keywords in the current file according to the setting on the Editor preferences page. Also, you can set a new option on the Editor preferences page to have the ABL Editor automatically adjust keyword casing as you type.
- A new command (Source>Correct Indentation, Ctrl+I) causes the ABL Editor to adjust indentation of lines in the current file.

Hot keys

Note the following changes and additions to default hot-key sequences for Progress OpenEdge:

<table>
<thead>
<tr>
<th>Hot Key Sequence</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALT+SHIFT+T</td>
<td>Add a static constructor to the current file.</td>
</tr>
<tr>
<td>ALT+SHIFT+Z, C</td>
<td>Surrounds the currently selected code with a <strong>CATCH</strong> block.</td>
</tr>
<tr>
<td>ALT+SHIFT+Z, F</td>
<td>Surrounds the currently selected code with a <strong>FINALLY</strong> block.</td>
</tr>
<tr>
<td>CTRL+SHIFT+F</td>
<td>Applies casing preferences to keywords in the current file. This sequence previously applied both keyword casing and indentation.</td>
</tr>
<tr>
<td>CTRL+I</td>
<td>(New) Adjusts indentation of lines in the current file.</td>
</tr>
<tr>
<td>CTRL+SHIFT+p</td>
<td>(New) Moves the cursor to the matching code element paired with the element at the current cursor position.</td>
</tr>
</tbody>
</table>

Accessing the Add Annotation wizard

The **Add Annotation** wizard is available by choosing **Source>Add Annotation** from the OpenEdge main menu bar. The wizard is no longer available by choosing **File>New>Other**.

If a file is open, you can access the **Add Annotation** wizard by choosing **Source>Add Annotation** from the context menu. You can also right-click in the **Progress Explorer** view and choose **OpenEdge>Add Annotation**.
See also
New and Noteworthy in Release 11.0 on page 28
New and Noteworthy in Release 10.2B on page 35
New and Noteworthy in Release 10.2A on page 38

New and Noteworthy in Release 10.2B

The 10.2B release of OpenEdge Architect (renamed as Progress Developer Studio for OpenEdge in Release 11.0) included new and revised features in the following areas:

- Upgrade to Eclipse 3.4.2 and Eclipse 3.5 support
- File layout changes
- New video demos
- Upgrade of OpenEdge Ultra Controls for .NET
- Update Assembly References tool
- Support for ABL language improvements
- ABL Editor context-filtered code assistance
- Improved ProBindingSource Designer
- Global Toolbox option
- Shared AVM
- Performance improvements
- AppServer support
- Class Browser enhancements
- Support for AppBuilder shortcut keys

Upgrade to Eclipse 3.4.2 and Eclipse 3.5 support

The OpenEdge Architect 10.2B release includes Eclipse 3.4.2. (The 10.2A release of OpenEdge Architect was shipped with Eclipse 3.4.) The upgrade to Eclipse 3.4.2 includes a new Welcome page that is more easily integrated with third-party plug-ins.

In addition, OpenEdge Architect 10.2B is certified to run on Eclipse 3.5.

File layout changes

In order to more fully support the Eclipse Equinox/p2 provisioning system for managing updates and installing new features, the file layout of OpenEdge Architect has been changed in Release 10.2B.

Briefly stated, the change involves the addition of several new directories that act as repositories to provision OpenEdge Architect and third-party plug-ins. The directory structure of OpenEdge-install-dir/oeide now looks similar to the following:
Notice that the `../oeide/architect` folder, which contained the installed OpenEdge Architect plugins in prior releases, no longer exists. OpenEdge Architect and all other installed plugins are now installed in `../oeide/eclipse/plugins`.

Also note that, in addition to the `Architect_repo` folder for OpenEdge Architect, there are two repository folders to provision third-party plugins. One is for third-party tools that have Eclipse 3.4.2 dependencies and the other is for third-party tools that have Eclipse 3.5.0 dependencies. OpenEdge Architect 10.2B supports both of these Eclipse platforms.

**New video demos**

Now there are a variety of demos, hosted on the Progress Communities Website, that illustrate the features of OpenEdge and OpenEdge Architect.

For demos describing how to get started with OpenEdge Architect, see OpenEdge Architect Videos: Getting Started Collection. There are also demos that show a variety of OpenEdge Architect features and these are located at OpenEdge Architect Videos: Using Architect Collection.

In addition, there are direct links to appropriate demos from various topics in the OpenEdge Architect online help.

**Upgrade of OpenEdge Ultra Controls for .NET**

Customers who license OpenEdge Ultra Controls for .NET receive version 9.2 of the controls with OpenEdge 10.2B. In that respect, OpenEdge 10.2B differs from OpenEdge 10.2A, which includes OpenEdge Ultra Controls version 8.1.

**Update Assembly References tool**

OpenEdge 10.2B includes a new Update Assembly References tool that helps users synchronize assembly references in existing projects with new versions of those assemblies. Each time a new version of a control set is installed, it is necessary to update these references. For example, the change from version 8.1 to version 9.2 of the OpenEdge Ultra Controls necessitates such an update.

You can invoke the Update Assembly References tool either from the OpenEdge > Tools > Migration menu or from a command prompt.

**Support for ABL language improvements**

OpenEdge 10.2B introduces ABL support for:
Abstract classes and data members

Strongly typed events

Accessing .NET generics (definition of new generics in ABL is not supported)

In OpenEdge Architect, code-generation wizards, code assistance, the Outline view, Quick Outline, and the Debugger have been extended to support these language improvements.

**ABL Editor context-filtered code assistance**

By default, code-completion assistance now initially offers only those proposals that are relevant in the context of the current source file. Users can press CTRL+SPACE to toggle the list between context-filtered suggestions and all suggestions. A preference setting is available to disable filtering of assistance proposals according to context.

**Improved ProBindingSource Designer**

In the Visual Designer, usability improvements have been made to the tool that helps users define the schema displayed by a data-bound control. Enhancements to the ProBindingSource Designer include support for drag-and-drop operations, streamlined support for importing a schema from a file or from a database, and an updated user interface.

**Global Toolbox option**

The Visual Designer now offers the option of sharing a common Toolbox among multiple projects. The option to use a dedicated Toolbox for a project remains available. A new project property setting controls this behavior for each project.

**Shared AVM**

In prior releases of OpenEdge Architect, each project in a workspace had its own instance of an ABL Virtual Machine (AVM). Basically, the AVM is a platform for compiling and running ABL code.

Release 10.2B of OpenEdge Architect introduces a shared AVM, which is a single AVM that can be used by any or all projects in a workspace. The shared AVM is useful if multiple projects require the same database connection, PROPATH, AppBuilder, assemblies, and AVM startup properties. The shared AVM is advantageous because it uses fewer memory and CPU resources.

To configure a shared AVM, select **Window > Preferences** from the main menu bar. Then open **OpenEdge Architect > Shared AVM** in the tree view of the Preferences dialog. You can set the AVM startup parameters on the Shared OpenEdge AVM preference page. In the child nodes, you can set the other properties that affect all the projects that use the shared AVM.

In the New OpenEdge Project wizard (**File > New > OpenEdge Project**), you can now choose whether you want to use an individual, project-specific AVM or a shared AVM. For existing projects in the workspace, you can go to the OpenEdge property page for the project and select the Use shared AVM option.

**Performance improvements**

The performance of OpenEdge Architect has been improved in the following areas:

- Updating the Outline view when changes occur
- Expanding include files
- Opening files
- Responding to user input
- Adding code sections (functions, methods, etc.) using the various source-editing wizards
AppServer support
New in Release 10.2B is support for the OpenEdge AppServer in OpenEdge Architect. With this support, you can now do the following:

• Manage AppServers from within OpenEdge Architect by defining connections to AppServer brokers.
• Use one tool—OpenEdge Architect—to perform multiple AppServer functions, such as starting or stopping servers or adding or trimming AppServer agents.

For additional AppServer-related tasks, you can use Progress Explorer, OpenEdge Management (with the required license), or OpenEdge Explorer. You can access OpenEdge Explorer and/or OpenEdge Management from within OpenEdge Architect.

• Create and publish an application to an AppServer, which allows you to test the application in an environment outside of the OpenEdge Architect workspace.
• Use the Eclipse launch configuration framework to set certain AppServer properties and create and store alternate configurations for each AppServer in your environment.
• Monitor AppServer broker and server status details through a server monitor view.

Class Browser enhancements
The Class Browser includes the following enhancements in 10.2B:

• The description of a class, method, property, or event now includes ABSTRACT, if applicable.
• The Class Browser view now displays .NET generic types.
• The Class Browser view recognizes the ABL events declared in an ABL class and populates them in the events section of class.
• The ABL Syntax section for an event, as viewed in the Summary pane, now provides sample code that you can use to define a method handler.
• The Class Browser uses the PROPATH specified at the project level when processing project classes. For a project using the shared AVM, the Class Browser shows the assemblies and the PROPATH defined for the shared AVM.

Support for AppBuilder shortcut keys
The Keys preference page (Window > Preferences > General > Keys) has a new choice for key bindings. You can select AppBuilder in the Scheme box to implement most of the AppBuilder shortcut keys.

See also
New and Noteworthy in Release 11.0 on page 28
Eclipse platform upgrade on page 43
Class Browser on page 48

New and Noteworthy in Release 10.2A
The 10.2A Release of OpenEdge Architect (renamed as Progress Developer Studio for OpenEdge in Release 11.0) included new and revised features in the following areas:

• Eclipse platform upgrade
• Visual Designer
• ABL Editor
Eclipse platform upgrade

The Eclipse platform for OpenEdge Architect 10.2A is Eclipse Release 3.4. The previous release of OpenEdge Architect (10.1C) ran on Eclipse 3.2.2. Changes implemented in Eclipse 3.3 and Eclipse 3.4 are documented on the Eclipse Web site.

Note: The Eclipse Software Update feature is completely new in release 3.4. A description of the 3.4 software update functionality is available from the Eclipse Web site. The new software update is available when you choose Help>Software Updates. Release 10.2A of OpenEdge Architect also supports the software update feature that existed in previous releases. It is available when you choose Help>Classic Software Updates.

Visual Designer

The Visual Designer, new in OpenEdge 10.2A, is a WYSIWIG editor for building user interfaces for OpenEdge applications. These interfaces are written in ABL but are based on .NET objects, with the result that the GUI is richer, more robust, and more contemporary in look and feel compared to a traditional OpenEdge GUI. The style of user interface built in ABL with .NET forms and controls is referred to as OpenEdge GUI for .NET.

Support for .NET objects in OpenEdge GUI clients is built into ABL. ABL developers can use .NET objects without resorting to another language such as C# or VB.NET. Because the OpenEdge GUI client hosts the Microsoft .NET runtime (CLR) within the AVM, .NET controls appear to the developer as if they were native ABL constructs. This powerful configuration not only allows .NET forms and controls in an OpenEdge application, but it also dispatches events from the .NET UI to event handling code written in ABL. Additionally, OpenEdge data objects can easily be used as a data source for a .NET control.

The Visual Designer is implemented as a plug-in to the Eclipse-based OpenEdge Architect. This provides an integrated development environment with quick access to other OpenEdge Architect tools such as Progress DB Navigator, the ABL Editor, and Tools for Business Logic.

ABL Editor

Launch configuration support

OpenEdge Architect now uses the launch framework in Eclipse to run or debug ABL programs.

Launching is defined as running or debugging a program from within Eclipse. A launch configuration is a set of options that affect how Eclipse runs a program. Launch configurations are useful because they allow you to create a number of different running and debugging environments to test your code.

You can use the Run Configurations and Debug Configurations dialogs to create custom launch configurations. Alternatively, you can use a default launch configuration derived from the workbench preferences and your program’s OpenEdge project settings.

Note: The Run as GUI application, Run as TTY application, Run Configuration, and Run Web options are no longer available on the main toolbar. Furthermore, the Run Configuration dialog, used in previous versions of OpenEdge Architect to create and start launch configurations, no longer exists. However, launch configurations created in previous versions of OpenEdge Architect migrate automatically to Eclipse launch configurations.

Array support

Array support in the ABL Editor now includes:
• An update to the New Class wizard so that return statements for generated methods or functions can return properly sized arrays.

• An update to the New Class wizard so that return statements for generated methods or functions can return properly sized arrays.

• The extension of Update Function Prototypes (a selection on the ABL Editor’s Source menu) to update the extent information for a function based on the signature of the actual function implementation.

• An update to the Outline view that indicates when a method or a function has an extent, and (optionally) the size of the extent.

• Support in context assistance help to display array information as part of the return type of a function, a method, or a property.

• An update to the Override/Implement Method dialog to display extent information in the labels for methods that return arrays.

Color coding in the Compare Editor

When you compare ABL files in the Eclipse Compare Editor, the files are color coded as they are color coded in the ABL Editor. Previously, ABL files appeared as plain text in the Eclipse Compare Editor.

To compare files in OpenEdge Architect, select two files in the Resources view. From the context menu, select Compare With>Each Other. (Note that you can compare three files when one file is the common ancestor of the other two. See the Eclipse Workbench User Guide for more information.)

Insertion point options when adding code

You now have the option to specify an insertion point when you add code prototypes with any of the following dialogs:

• Add Procedure

• Add Function

• Add Method

• Add Property

The Insertion position field on these dialogs gives you the following options:

• **Alphabetical order** - Insert a code prototype based on its name. Case is ignored.

**Note**: This feature does not sort the procedures, functions, methods, or properties that already exist in the file. If they are already ordered alphabetically by name, insertion of new code is done at the correct position in the list. Otherwise, insertion is the first valid alphabetic position found during a top to bottom lookup.

• **Cursor position** - Insert code prototype at the current position of the cursor in the file.

• **First/Last** - Insert the code prototype as the first or last item.

• **After name** - Insert the code prototype after the named procedure.

Enhancements to code-completion assistance

Code-completion assistance gives you suggestions for completing the code that you are typing in the ABL Editor. To get code-completion assistance, press CTRL+SPACE.
Code-completion assistance is improved to filter suggestions based on context. Code-completion assistance for keywords is improved to show relevant keywords only. The older scheme of code-completion is retained and you can access it by pressing CTRL+SPACE again (Repeatedly pressing CTRL+SPACE toggles between context filtered suggestions and all suggestions).

Code completion assistance is also enhanced to provide suggestions for completing:

- Object variable and property names that are reserved keywords
- Object references for the CREATE statement
- Procedures names and paths for the RUN statement (based on PROPATH)
- Widget variables and handles
- Parameters of procedures and methods
- Fields for temp-tables, buffers, and datasets
- Preprocessor names
- Subscripted array references

**Tabular formatting of code**

The tabular formatting feature allows you to format code in columns. For example, the following shows code before tabular formatting:

```plaintext
DEFINE VARIABLE aChar1 AS CHARACTER NO-UNDO.
DEFINE VARIABLE aChar22 AS CHARACTER NO-UNDO.
DEFINE VARIABLE aChar333 AS CHARACTER NO-UNDO.
```

After tabular formatting, the code is much easier to scan as shown in the following:

```plaintext
DEFINE VARIABLE aChar1 AS CHARACTER NO-UNDO.
DEFINE VARIABLE aChar222 AS CHARACTER NO-UNDO.
DEFINE VARIABLE aChar333 AS CHARACTER NO-UNDO.
```

Only the following statements support tabular formatting:

- DEFINE VARIABLE
- DEFINE TEMP-TABLE
- DEFINE BUFFER
- ASSIGN

Select Source>Correct Indentation from the main menu bar to format the currently active ABL file. If no code is selected, tabular formatting applies to every supported statement in the file.

Tabular formatting is enabled by default. To disable tabular formatting, select Window>Preferences>OpenEdge Architect>Editor from the main menu bar. Clear the Enable tabular formatting check box.

**Support for adding properties and methods to interface files**

You can now use the Add Methods (ALT+SHIFT+M) and Add Properties (ALT+SHIFT+Y) dialogs on interface files. However some of the options in these dialogs are disabled because they do not apply to interface files.
The following are disabled in the Add Properties dialog:

• All modifiers except Public
• Initial value field
• Insert implementation check boxes

The following are disabled in the Add Methods dialog:

• All modifiers except Public
• All code generation options

Quick Outline

The Quick Outline is a pop-up window that shows a tree view of the ABL code file that currently has focus. You can click on a node in the tree view to navigate to a particular section of the file.

It is essentially the same as the Outline view but it is more convenient. Select Navigate>Quick Outline or press CTRL+O to start Quick Outline. (In class files, pressing CTRL+O repeatedly toggles between members from the current class and inherited members.)

Viewer for pre-processed code

The new Preprocessor view shows code that is obtained after all the preprocessors (include files, preprocessor definitions and logic) are expanded. It is a read-only view. Syntax coloring and highlighting are applied to the code in the Preprocessor view so it appears the same as the code in the ABL Editor.

The Preprocessor view does not appear in the default OpenEdge Editor perspective. You can access it by selecting Window>Show View>Preprocessor View on the main menu bar. It can also be started from the ABL Editor's context menu by selecting Show In>Preprocessor View.

Class Browser

With this release, the Class Browser extends its support to .NET classes, allowing you to see details about all your .NET resources, namespaces, and types. The details you see come from the assemblies that are associated with the workspace and are also available for any external resources you add to the Class Browser.

The Class Browser displays type information for .NET classes using summary text and ABL syntax.

The Class Browser also lets you see extent information for methods, properties, and variables that are arrays. This information appears in the Content pane and the Summary pane.

The Class Browser toolbar now includes Back and Forward buttons that let you navigate through previously selected items.

Debugger

The OpenEdge Architect Debugger includes the following enhancements in this release:

• You can set two preferences for your Debugger environment:
  • A connection time-out value, which establishes the time-out in milliseconds for the Debugger to wait when connecting to an OpenEdge client
  • A property evaluation option, which enables the evaluation of properties with accessors

• In addition to setting up property evaluation by choosing the option on the Debug preferences page, you can alternatively set up the evaluation of properties with accessors by doing either of the following:
  • Using the Evaluate Property menu option in the Variables and Expressions views
  • Adding the property to the Expressions view. The property is then evaluated automatically
• You can define launch configurations for debugging.
• You can use a class file (.cls), as well as a .p or .w file, as a starting program for the Debugger.
• In the Variables view, you can:
  • Opt to show properties
  • View the actual type of an object reference
  • Expand object references
• In the Expressions view, you can expand object references, temp-tables, handles, arrays, and datasets.
• In an Editor window, you can hover over a variable to see the variable’s value as you debug.

**Progress XML Tools**

The 10.2A release of OpenEdge Architect includes the following Progress XML tools:

• **XML editor** - An editor for creating and modifying Extensible Markup Language (XML) files
• **XML Schema editor** - An editor for generating XML schema from an XML document
• **XPath Helper** - A tool for generating and testing XPath expressions
• **WSDL editor** - An editor for creating and modifying Web Services Description Language (WSDL) files

You can start the editors by choosing File>New>Other>Progress XML Editors from the main menu bar. You can launch the XPath Helper from several different locations. For example, you can launch it from the source page of the XML editor. You can also launch it from other editors, at locations where you specify XPath expressions.

**See also**

- New and Noteworthy in Release 11.0 on page 28
- New and Noteworthy in Release 10.2B on page 35
- New and Noteworthy in Release 10.2C on page 32

**Eclipse platform upgrade**

The 10.2A Release of OpenEdge Architect (renamed as Progress Developer Studio for OpenEdge in Release 11.0) included new and revised features in the following areas:

• **Eclipse platform upgrade**
• **Visual Designer**
• **ABL Editor**
• **Class Browser**
• **Debugger**
• **Progress XML Tools**

**Eclipse platform upgrade**

The Eclipse platform for OpenEdge Architect 10.2A is Eclipse Release 3.4. The previous release of OpenEdge Architect (10.1C) ran on Eclipse 3.2.2. Changes implemented in **Eclipse 3.3** and **Eclipse 3.4** are documented on the Eclipse Web site.
The Eclipse Software Update feature is completely new in release 3.4. A description of the 3.4 software update functionality is available from the Eclipse Web site. The new software update is available when you choose Help > Software Updates. Release 10.2A of OpenEdge Architect also supports the software update feature that existed in previous releases. It is available when you choose Help > Classic Software Updates.

Visual Designer

The Visual Designer, new in OpenEdge 10.2A, is a WYSIWIG editor for building user interfaces for OpenEdge applications. These interfaces are written in ABL but are based on .NET objects, with the result that the GUI is richer, more robust, and more contemporary in look and feel compared to a traditional OpenEdge GUI. The style of user interface built in ABL with .NET forms and controls is referred to as OpenEdge GUI for .NET.

Support for .NET objects in OpenEdge GUI clients is built into ABL. ABL developers can use .NET objects without resorting to another language such as C# or VB.NET. Because the OpenEdge GUI client hosts the Microsoft .NET runtime (CLR) within the AVM, .NET controls appear to the developer as if they were native ABL constructs. This powerful configuration not only allows .NET forms and controls in an OpenEdge application, but it also dispatches events from the .NET UI to event handling code written in ABL. Additionally, OpenEdge data objects can easily be used as a data source for a .NET control.

The Visual Designer is implemented as a plug-in to the Eclipse-based OpenEdge Architect. This provides an integrated development environment with quick access to other OpenEdge Architect tools such as Progress DB Navigator, the ABL Editor, and Tools for Business Logic.

ABL Editor

The ABL Editor has the following new and revised features in the 10.2A release:

- Launch configuration support
- Array support
- Color coding in the Compare Editor
- Insertion point options when adding code
- Enhancements to code-completion assistance
- Tabular formatting of code
- Quick Outline
- Support for adding properties and methods to interface files
- Viewer for pre-processed code

Launch configuration support

OpenEdge Architect now uses the launch framework in Eclipse to run or debug ABL programs.

Launching is defined as running or debugging a program from within Eclipse. A launch configuration is a set of options that affect how Eclipse runs a program. Launch configurations are useful because they allow you to create a number of different running and debugging environments to test your code.

You can use the Run Configurations and Debug Configurations dialogs to create custom launch configurations. Alternatively, you can use a default launch configuration derived from the workbench preferences and your program's OpenEdge project settings.

The Run as GUI application, Run as TTY application, Run Configuration, and Run Web options are no longer available on the main toolbar. Furthermore, the Run Configuration dialog, used in previous versions of OpenEdge Architect to create and start launch configurations, no longer exists. However, launch configurations created in previous versions of OpenEdge Architect migrate automatically to Eclipse launch configurations.
Array support

Array support in the ABL Editor now includes:

- An update to the New Class wizard so that return statements for generated methods or functions can return properly sized arrays.
- A new Extent choice in the Add Function, Add Method, and Add Property dialogs, where you can optionally specify the size of the extent. (Note that Extent is disabled for AppBuilder files.)
- The extension of Update Function Prototypes (a selection on the ABL Editor's Source menu) to update the extent information for a function based on the signature of the actual function implementation.
- An update to the Outline view that indicates when a method or a function has an extent, and (optionally) the size of the extent.
- Support in context assistance help to display array information as part of the return type of a function, a method, or a property.
- An update to the Override/Implement Method dialog to display extent information in the labels for methods that return arrays.

Color coding in the Compare Editor

When you compare ABL files in the Eclipse Compare Editor, the files are color coded as they are color coded in the ABL Editor. Previously, ABL files appeared as plain text in the Eclipse Compare Editor.

To compare files in OpenEdge Architect, select two files in the Resources view. From the context menu, select Compare With > Each Other. (Note that you can compare three files when one file is the common ancestor of the other two. See the Eclipse Workbench User Guide for more information.)

Insertion point options when adding code

You now have the option to specify an insertion point when you add code prototypes with any of the following dialogs:

- Add Procedure
- Add Function
- Add Method
- Add Property

The Insertion position field on these dialogs gives you the following options:

- **Alphabetical order** - Insert a code prototype based on its name. Case is ignored.

**Note:** This feature does not sort the procedures, functions, methods, or properties that already exist in the file. If they are already ordered alphabetically by name, insertion of new code is done at the correct position in the list. Otherwise, insertion is the first valid alphabetic position found during a top to bottom lookup.

- **Cursor position** - Insert code prototype at the current position of the cursor in the file.
- **First/Last** - Insert the code prototype as the first or last item
- **After name** - Insert the code prototype after the named procedure.

Enhancements to code-completion assistance

Code-completion assistance gives you suggestions for completing the code that you are typing in the ABL Editor. To get code-completion assistance, press CTRL+SPACE.
Code-completion assistance is improved to filter suggestions based on context. Code-completion assistance for keywords is improved to show relevant keywords only. The older scheme of code-completion is retained and you can access it by pressing `CTRL+SPACE` again (Repeatedly pressing `CTRL+SPACE` toggles between context filtered suggestions and all suggestions).

Code completion assistance is also enhanced to provide suggestions for completing:

- Object variable and property names that are reserved keywords
- Object references for the CREATE statement
- Procedures names and paths for the RUN statement (based on PROPATH)
- Widget variables and handles
- Parameters of procedures and methods
- Fields for temp-tables, buffers, and datasets
- Preprocessor names
- Subscripted array references

**Tabular formatting of code**

The tabular formatting feature allows you to format code in columns. For example, the following shows code before tabular formatting:

```
DEFINE VARIABLE aChar1 AS CHARACTER NO-UNDO.
DEFINE VARIABLE aaChar2 AS CHARACTER NO-UNDO.
DEFINE VARIABLE aaaChar3 AS CHARACTER NO-UNDO.
```

After tabular formatting, the code is much easier to scan as shown in the following:

```
DEFINE VARIABLE aChar1 AS CHARACTER NO-UNDO.
DEFINE VARIABLE aaChar2 AS CHARACTER NO-UNDO.
DEFINE VARIABLE aaaChar3 AS CHARACTER NO-UNDO.
```

Only the following statements support tabular formatting:

- `DEFINE VARIABLE`
- `DEFINE TEMP-TABLE`
- `DEFINE BUFFER`
- `ASSIGN`

Select **Source > Correct Indentation** from the main menu bar to format the currently active ABL file. If no code is selected, tabular formatting applies to every supported statement in the file.

Tabular formatting is enabled by default. To disable tabular formatting, select **Window > Preferences > OpenEdge Architect > Editor** from the main menu bar. Clear the Enable tabular formatting check box.

**Quick Outline**

The Quick Outline is a pop-up window that shows a tree view of the ABL code file that currently has focus. You can click on a node in the tree view to navigate to a particular section of the file.

It is essentially the same as the Outline view but it is more convenient. Select **Navigate > Quick Outline** or press `CTRL+O` to start Quick Outline. (In class files, pressing `CTRL+O` repeatedly toggles between members from the current class and inherited members.)

**Support for adding properties and methods to interface files**

You can now use the **Add Methods** (ALT+SHIFT+M) and **Add Properties** (ALT+SHIFT+Y) dialogs on interface files. However some of the options in these dialogs are disabled because they do not apply to interface files.
The following are disabled in the Add Properties dialog:

- All modifiers except Public
- Initial value field
- Insert implementation check boxes

The following are disabled in the Add Methods dialog:

- All modifiers except Public
- All code generation options

**Viewer for pre-processed code**

The new Preprocessor view shows code that is obtained after all the preprocessors (include files, preprocessor definitions and logic) are expanded. It is a read-only view. Syntax coloring and highlighting are applied to the code in the Preprocessor view so it appears the same as the code in the ABL Editor.

The Preprocessor view does not appear in the default OpenEdge Editor perspective. You can access it by selecting **Window > Show View > Preprocessor View** on the main menu bar. It can also be started from the ABL Editor's context menu by selecting **Show > In > Preprocessor View**.

**Class Browser**

With this release, the Class Browser extends its support to .NET classes, allowing you to see details about all your .NET resources, namespaces, and types. The details you see come from the assemblies that are associated with the workspace and are also available for any external resources you add to the Class Browser.

The Class Browser displays type information for .NET classes using summary text and ABL syntax.

The Class Browser also lets you see extent information for methods, properties, and variables that are arrays. This information appears in the Content pane and the Summary pane.

The Class Browser toolbar now includes Back and Forward buttons that let you navigate through previously selected items.

**Debugger**

The OpenEdge Architect Debugger includes the following enhancements in this release:

- You can set two preferences for your Debugger environment:
- A connection time-out value, which establishes the time-out in milliseconds for the Debugger to wait when connecting to an OpenEdge client
- A property evaluation option, which enables the evaluation of properties with accessors
- In addition to setting up property evaluation by choosing the option on the Debug preferences page, you can alternatively set up the evaluation of properties with accessors by doing either of the following:
  - Using the **Evaluate Property** menu option in the Variables and Expressions views.
  - Adding the property to the Expressions view. The property is then evaluated automatically.
- You can define launch configurations for debugging.
- You can use a class file (.cls), as well as a .p or .w file, as a starting program for the Debugger.
- In the Variables view, you can:
  - Opt to show properties
  - View the actual type of an object reference
  - Expand object references
• In the Expressions view, you can expand object references, temp-tables, handles, arrays, and datasets.
• In an Editor window, you can hover over a variable to see the variable's value as you debug.

**Progress XML Tools**
The 10.2A release of OpenEdge Architect includes the following Progress XML tools:
• **XML editor** - An editor for creating and modifying Extensible Markup Language (XML) files.
• **XML Schema editor** - An editor for generating XML schema from an XML document.
• **XPath Helper** - A tool for generating and testing XPath expressions.
• **WSDL editor** - An editor for creating and modifying Web Services Description Language (WSDL) files.

You can start the editors by choosing *File > New > Other > Progress XML Editors* from the main menu bar. You can launch the XPath Helper from several different locations. For example, you can launch it from the source page of the XML editor. You can also launch it from other editors, at locations where you specify XPath expressions.

**See also**
The Progress Developer Studio for OpenEdge Guide on page 28
Help display options on page 35
Context sensitive help on page 48

**Class Browser**

This section describes the new features and changes that were implemented in the 10.1C release of OpenEdge Architect (renamed as Progress Developer Studio for OpenEdge in Release 11.0).

**Class Browser**

A new view called the **Class Browser** allows you to see the structure and content of the various classes that you work with in creating ABL applications. The Class Browser's contents are organized by resources, which are collections of classes (types).

**Progress DB Navigator**

In order to make it more independent, the OpenEdge DB Navigator plug-in has been slightly restructured and was renamed the Progress DB Navigator. You will notice a change in the location and structure of its Preferences as well as a renaming of the perspective. In addition to continuing to provide the driver for the OpenEdge database, Progress Software now provides Type IV DataDirect JDBC drivers for the following databases: ORACLE, Microsoft SQL Server, IBM DB2, Sybase, and Informix.

**ABL Editor**

**Support for structured error handling**

To support ABL enhancements in error-handling functionality, OpenEdge Architect wizards, menus, and context assistance now provide options for automatically generating:

• Routine-level error-handling statements in new classes and procedures
• CATCH and FINALLY blocks in new or existing source files

**Performance-related options for code assistance**
The Editor Assistance preferences page includes several new options that can mitigate adverse impact on system response time caused by waiting for the code to be analyzed:

- You can specify the maximum amount of time that OpenEdge Architect spends updating code analysis before responding to requests for code-completion assistance. If the re-parsing of the code is not completed within the specified interval (by default 300 milliseconds), OpenEdge Architect proposes completion options based on the most recent full analysis.

- You can disable highlighting of matching elements. That is, you can suppress the display of a marker at the position of the code element, such as a parenthesis or a beginning or ending statement in a code block, that is paired with the element at the current cursor position.

- You can suppress field descriptions in completion proposals for database tables.

Using these options can improve performance, particularly when you work with very large source files or remote databases.

**Support for static data members, methods, and properties**

To support the use of the STATIC keyword where applicable, OpenEdge Architect macros, wizards, and menus now provide options for automatically defining data members, methods, and properties as static.

In the Outline view, the icons for static data members, methods, and properties appear with a red "S" overlaying the standard image.

Improved support for coding methods implemented via interface

The New Class wizard has been enhanced to automatically generate method stubs for all methods implemented via interface. In addition, the former Override Methods dialog, now called Override/Implement Methods, has been enhanced to generate stubs for interface methods on demand.

**Enhanced macro functionality**

OpenEdge Architect now uses standard Eclipse template functionality for creating and managing macros (also called templates). Procedures for using and managing macros are largely unchanged from previous versions, but the enhanced design offers these advantages:

- You can now define and run multiple-line macros. Previously, macros were restricted to a single line.

- For creating custom macros, the New Template editor includes an Insert Variable command that improves usability.

To make custom macros created in previous versions of OpenEdge Architect available for use in version 10.1C, migration is required. When you first select a workspace created in an earlier version, OpenEdge Architect offers the option of migrating the macros for that workspace. If you decline to perform the migration at that time, you can do so at your convenience by selecting OpenEdge > Migration > Migrate 10.1A or 10.1B Macros to Templates.

**Formatting changes**

The following changes have been made to OpenEdge Architect formatting and code-assistance options:

- The Format option no longer appears on the Source menu, its functions replaced by separate commands as described in the next two entries in this list.

- A new command (Source > Correct Case, Ctrl+Shift+F) causes the ABL Editor to adjust the case of keywords in the current file according to the setting on the Editor preferences page. Also, you can set a new option on the Editor preferences page to have the ABL Editor automatically adjust keyword casing as you type.

- A new command (Source > Correct Indentation, Ctrl+I) causes the ABL Editor to adjust indentation of lines in the current file.
Hot keys

Note the following changes and additions to default hot-key sequences for OpenEdge Architect:

<table>
<thead>
<tr>
<th>Hotkey Combination</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALT+SHIFT+T</td>
<td>Adds a static constructor to the current file.</td>
</tr>
<tr>
<td>ALT+SHIFT+Z, C</td>
<td>Surrounds the currently selected code with a CATCH block.</td>
</tr>
<tr>
<td>ALT+SHIFT+Z, F</td>
<td>Surrounds the currently selected code with a FINALLY block.</td>
</tr>
<tr>
<td>CTRL+SHIFT+F</td>
<td>Applies casing preferences to keywords in the current file. This sequence previously applied both keyword casing and indentation.</td>
</tr>
<tr>
<td>CTRL+I</td>
<td>(New) Adjusts indentation of lines in the current file.</td>
</tr>
<tr>
<td>CTRL+SHIFT+P</td>
<td>(New) Moves the cursor to the matching code element paired with the element at the current cursor position.</td>
</tr>
</tbody>
</table>

Accessing the Add Annotation wizard

The Add Annotation wizard is available by choosing Source > Add Annotation from the OpenEdge Architect main menu bar. The wizard is no longer available by choosing File > New > Other.

If a file is open, you can access the Add Annotation wizard by choosing Source > Add Annotation from the context menu. You can also right-click in the Resources view and choose OpenEdge > Add Annotation.

See also

- New and Noteworthy in Release 11.0 on page 28
- What's New in Release 10.2B on page 35
- Eclipse platform upgrade on page 43

Eclipse terminology

Eclipse is an open, extensible development environment that supports a wide range of tools and applications from diverse sources. All applications are installed as plug-ins to the Eclipse environment. You can read about and obtain many such plug-ins at http://www.eclipseplugincentral.com.

While running Eclipse, you have access to its extensive online documentation. You can refer to the Eclipse Workbench User Guide, accessible from Help > Help Contents. Also see Eclipse information resources for links to external sources of Eclipse information.

Workspaces

A workspace is a folder where your work is stored. A workspace can contain almost any number of projects, which are usually structured as sub-folders of the workspace folder. When you start Progress Developer Studio for OpenEdge, Eclipse prompts you for the path name of your workspace. You can enter either an existing or a new folder.

Projects and resources

An important fundamental concept is that Eclipse is a project-based environment. Projects contain resources, which are the files and folders that are organized under a particular project name. You can create resources within a project, import them from another project, or add them from your file system.

Views and editors
Eclipse presents information in windows called views, each of which serves a specific purpose. For example, the Outline view shows the logical structure of a resource, such as an ABL source code file or an application model. The Database Structure view allows you to examine a database schema.

In addition to views, each perspective usually includes an editing area. You can work with multiple files of different types simultaneously in this area, using the appropriate editor for each file according to its type.

**Perspectives**

In the Eclipse Workspace, you can arrange multiple views and save that set of views as a perspective. Progress Developer Studio for OpenEdge offers several predefined perspectives integrating diverse tools that combine to support a broader function, such as code editing, debugging, or database design.

**See also**

- Working with OpenEdge perspectives on page 62
- OpenEdge perspectives and views on page 77

**Preferences and properties**

In Eclipse, the distinction between preferences and properties is generally a matter of scope. Preferences apply to the entire workspace, while properties apply to individual resources.

Preferences usually apply either to the Eclipse framework itself or to individual plug-ins. To access preference dialogs, select **Window > Preferences** on the main menu bar. From the preference dialogs, you can access pages containing general Eclipse settings as well as pages for individual plug-ins. For example, there are preferences that apply to all the editors that are installed in the Eclipse workbench. There are also preferences that apply only to specific editors (such as the ABL Editor, the Visual Designer, and XML editors).

Properties tend to have a narrower scope. They apply to individual resources (projects, folders, or files) in a workspace. To access property sheets, right-click on a selected resource in a Resources view and select Properties from the context menu.

Note: When an OpenEdge project uses the shared AVM, some project properties must be set at the workspace level. These properties (AppBuilder, Assemblies, Database Connections, Projects, and PROPATH) are identical for all projects that use the shared AVM. Set workspace-level properties from dialogs under the **Shared OpenEdge AVM preference** page from the main menu bar (**Window > Preferences > Progress OpenEdge > Shared AVM**).

**See also**

- Eclipse Terminology on page 50
- The ABL Virtual Machine AVM on page 52
- Setting Workspace preferences on page 61
- Preferences on page 65
- Shared AVM preference page on page 68
Database connection profiles

<table>
<thead>
<tr>
<th>Demo</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Defining Workspace Database Connections</strong></td>
</tr>
</tbody>
</table>

**Note:** The OpenEdge Release 10.2B demos have not been updated for OpenEdge Release 11.0. Since 10.2B, OpenEdge Architect has been renamed as Progress Developer Studio for OpenEdge. In addition, OpenEdge Release 11.0 includes some feature enhancements and other changes. Despite the differences between 10.2B and 11.0, the demos are still a useful introduction to Progress Developer Studio for OpenEdge.

Database connection profiles contain all the information necessary to connect to a database. The information can include startup parameters, user name and password, host name, port number, and more. Connection profiles are stored in an XML file (databaseConnection.xml), which is part of the metadata of a workspace.

Database connection profiles are available to all projects in a workspace. You can select any or all of the profiles from a project's Database Connections property sheet.

You can transfer the database connection profiles to other workspaces using Export/Import utilities in Progress Developer Studio for OpenEdge.

**See also**
- Database preferences on page 67
- Add/Edit Connection Profile wizard on page 72

The ABL Virtual Machine (AVM)

Basically, the AVM is a platform for compiling and running ABL code. Often called the OpenEdge runtime or client, the AVM also provides many other services, such as performing syntax checks, running OpenEdge and user-designed tools, and running startup procedures. In Progress Developer Studio for OpenEdge, every OpenEdge project is associated with an AVM.

**Note:** The AVM process is `prowin32.exe` for Windows applications and `_progres.exe` for TTY applications.

Project-specific AVMs

You might want to configure individual AVMs when projects in a workspace do not have the same requirements. You could, for example, create projects with different PROPATH settings by configuring an individual AVM for each project.

Configure project-level AVMs on the properties pages for OpenEdge projects (**Project > Properties > Progress OpenEdge**).

Shared AVMs

You can configure an AVM that is available to every project within a workspace. If, for example, you are developing an application that is divided into multiple project modules within a single workspace, you might want all the project modules to use an AVM that has the same startup parameters, the same PROPATH, the same database connections and so on. Configuring a shared AVM for all the project modules guarantees that all those settings are the same. In addition, the shared AVM conserves memory and CPU resources.
You configure a shared AVM as a workspace preference (Window > Preferences > Progress OpenEdge > Shared AVM).

Runtime AVMs

By default, when you run an ABL executable in a project, the AVM associated with the project (project-specific or shared) runs the ABL executable. However, you can also create launch configurations that start a new instance of the AVM when you run the file.

Configure separate runtime AVMs when you create launch configurations in your workspace (Run > Run Configurations...).

Note: You cannot use a shared or project-level AVM when you are running in debug mode. You must configure a separate runtime AVM (Run > Debug Configurations...).

See also

Preferences and properties on page 51
Setting the default AVM startup parameters on page 61
Setting up a shared AVM on page 62
Shared AVM preference page on page 68
Startup preference page on page 66

How to get started with Progress Developer Studio for OpenEdge

Getting started with Progress Developer Studio for OpenEdge requires some basic knowledge of the Eclipse framework. At a minimum, you should understand the following concepts that are explained in the Eclipse Workbench User Guide:

• Projects
• Views
• Perspectives

Also see the Basic Tutorial in the Eclipse Workbench User Guide for more information about working in the Eclipse framework.

See the following help topics when you are ready to begin using Progress Developer Studio for OpenEdge:

• Starting Progress Developer Studio for OpenEdge on page 59
• Setting workspace preferences on page 61
• Working with OpenEdge perspectives on page 62
• Getting help on page 54

Once Progress Developer Studio for OpenEdge is running, either open an existing project or create a new project.

See also

Eclipse terminology on page 50
Preferences and properties on page 51
Database connection profiles on page 52
The ABL Virtual Machine AVM on page 52
Starting Progress Developer Studio for OpenEdge on page 59
Getting help

This section describes how to locate and use the help that is available for Progress Developer Studio for OpenEdge.

Help display options

To change the way help is displayed, select Windows > Preferences from the main menu bar of Progress Developer Studio for OpenEdge. Select the Help node in the tree view. The various options for displaying help appear.

For example, you can choose to display help in an external browser rather than in the Eclipse help browser. Displaying help in an external browser is useful when you are working in a modal window in Eclipse. If you are viewing help in the default Eclipse browser, you cannot view help when the modal window is open. If you display help in an external browser, you can change focus from the modal window to the browser.

For more information about help display options, see Help preferences in the Eclipse Workbench User Guide.

See also

The Progress Developer Studio for OpenEdge Guide on page 54
Searching on page 55
Context sensitive help on page 56
Context sensitive help for NE controls Windows only on page 56
Visual Designer tutorials on page 57
Cheat sheets on page 57
Demos on page 57
OpenEdge information resources on page 58
Eclipse information resources on page 59

The Progress Developer Studio for OpenEdge Guide

The Progress Developer Studio for OpenEdge Guide, an online help volume in the Eclipse help system, is the primary source of documentation for the tools in Progress Developer Studio.

The content of the Progress Developer Studio for OpenEdge Guide is organized under the following topics:

• Getting Started - Contains help on basic topics, such as starting Progress Developer Studio for OpenEdge setting up a workspace, working with perspectives, and getting help.
• OpenEdge Projects - Describes how to set up and use OpenEdge projects.
• AppServer - Describes AppServer support in Progress Developer Studio for OpenEdge.
• OpenEdge REST - Describes REST support in Progress Developer Studio for OpenEdge.
• OpenEdge Data Object - Describes Data Object support in Progress Developer Studio for OpenEdge.
• OpenEdge Business Rules - Describes OpenEdge Business Rules in Progress Developer Studio for OpenEdge.
• WebSpeed - Describes WebSpeed support in Progress Developer Studio for OpenEdge.
• ABL Editor - Contains help on using the ABL code editor.
• Visual Designer - Contains help on developing graphical user interfaces with .NET forms and controls.
• Class Browser - Contains help on an Progress Developer Studio for OpenEdge view that allows you to see the structure and content of the various classes that you work with in creating ABL applications. The Class Browser's contents are organized by resources, which are collections of classes (types).
• Running and Debugging ABL Programs - Contains help on using launch configurations and the ABL Debugger.
• GUI Designer - Describes how to launch and run non-Eclipse OpenEdge tools (such as the AppBuilder and the Data Dictionary).
• Meta Catalog - Contains help on a tool for indexing and finding code elements.
• Tools for Business Logic - Contains help on tools for modeling application components.
• Customization - Describes how to modify the Progress Developer Studio for OpenEdge user interface, how to create automated startup procedures, and how to subscribe to events.
• ABL Language Reference - An online version of the OpenEdge Development: ABL Reference.

In addition to the Progress Developer Studio for OpenEdge Guide, the Progress DB Navigator Guide provides help on a tool you can use to manage, explore, and connect to databases.

Help display options
Searching
Context-sensitive help
Context-sensitive help for .NET controls (Windows only) on page 56
Visual Designer tutorials on page 57

Cheat sheets
Demos
OpenEdge information resources
Eclipse information resources

Searching
The default Eclipse help browser allows you to limit text searches to specified modules in the help system. Use the Search scope link at the top of browser to define a named search scope.

You can also modify your searches by creating query statements or by expanding or limiting results. See Searching help in the Eclipse Workbench User Guide.

Also note that you can do an index search. On the Eclipse help browser, click on the index icon at the bottom of the Contents pane. The Index pane, which allows you to either search or browse the index, replaces the Contents pane.

See also
The Progress Developer Studio for OpenEdge Guide on page 54
Help display options on page 54
Context sensitive help on page 56
Context-sensitive help

To access the help for a particular user-interface element (view, editor, or dialog):

1. Change focus to the user-interface element.

2. Press F1. Or, in some cases, click on a help icon 📚.

A list of topics appears either in an InfoPop or in a Help view. (See Help preferences in the Eclipse Workbench User Guide for information.) In either case, you see a primary list of topics at the top. At the bottom, you can access dynamic help which is a list of topics generated by a search on the name of the widget and the name of the perspective.

If you want to see the Progress Developer Studio for OpenEdge help topics in the primary list of available topics, select an element within the widget (for example, a node in a tree view or a text field). Otherwise, you might see general Eclipse help in the primary list with the Progress Developer Studio for OpenEdge topics appearing in the dynamic help list.

See also
The Progress Developer Studio for OpenEdge Guide on page 54
Help display options on page 54
Searching on page 55
Context sensitive help for NET controls Windows only on page 56
Visual Designer tutorials on page 57
Cheat sheets on page 57
Demos on page 57
OpenEdge information resources on page 58
Eclipse information resources on page 59

Context-sensitive help for .NET controls (Windows only)

When you are working in the Visual Designer, you can get help on an individual control by pressing F1 when a control is selected on the Design Canvas.

See also
The Progress Developer Studio for OpenEdge Guide on page 54
Help display options on page 54
Searching on page 55
Context sensitive help on page 56
Visual Designer tutorials on page 57
Cheat sheets on page 57
Demos on page 57
OpenEdge information resources on page 58
Eclipse information resources on page 59
Visual Designer tutorials

You can find tutorials that explain the features and functions of the OpenEdge Visual Designer in OpenEdge Getting Started: Introducing the Progress Developer Studio for OpenEdge Visual Designer, which is available from OpenEdge Product Documentation on the Progress Communities Web site.

Note: The Visual Designer runs only on Windows platforms.

See also
The Progress Developer Studio for OpenEdge Guide on page 54
Help display options on page 54
Searching on page 55
Context sensitive help on page 56
Context sensitive help for NET controls Windows only on page 56
Cheat sheets on page 57
Demos on page 57
OpenEdge information resources on page 58
Eclipse information resources on page 59

Cheat sheets

The Eclipse framework provides cheat sheets as guides for completing complex procedures. This release provides a number of cheat sheets that supplement the procedural help for Progress Developer Studio for OpenEdge. Choose Help > Cheat Sheets from the main menu bar to view the list of available cheat sheets.

See also
The Progress Developer Studio for OpenEdge Guide on page 54
Help display options on page 54
Searching on page 55
Context sensitive help on page 56
Context sensitive help for NET controls Windows only on page 56
Visual Designer tutorials on page 57
Demos on page 57
OpenEdge information resources on page 58
Eclipse information resources on page 59

Demos

There are a variety of demos, hosted on the PSDN website, that illustrate the features of OpenEdge and Progress Developer Studio for OpenEdge.

Note: The OpenEdge Release 10.2B demos have not been updated for OpenEdge Release 11.0. Since 10.2B, OpenEdge Architect has been renamed as Progress Developer Studio for OpenEdge. In addition, OpenEdge Release 11.0 includes some feature enhancements and other changes. Despite the differences between 10.2B and 11.0, the demos are still a useful introduction to Progress Developer Studio for OpenEdge.
For demos describing how to get started with Progress Developer Studio for OpenEdge, see OpenEdge Architect Videos: Getting Started Collection. There are also demos that show a variety of Progress Developer Studio for OpenEdge features at OpenEdge Architect Videos: Using Architect Collection.

For general information about both OpenEdge and Progress Developer Studio for OpenEdge, see the OpenEdge Tour.

See also
The Progress Developer Studio for OpenEdge Guide on page 54
Help display options on page 54
Searching on page 55
Context sensitive help on page 56
Context sensitive help for NET controls Windows only on page 56
Visual Designer tutorials on page 57
Cheat sheets on page 57
Demos on page 57
OpenEdge information resources on page 58
Eclipse information resources on page 59

OpenEdge information resources
There is a lot of useful information about OpenEdge available at the following Web sites:

OpenEdge Community
http://communities.progress.com/pcom/community/psdn/openedge
http://communities.progress.com/pcom/community/psdn/openedge

OpenEdge Architect (Progress Developer Studio for OpenEdge) forum
http://communities.progress.com/pcom/community/psdn/openedge/architect

OpenEdge Product Tour and Progress Software Evaluation Kits
http://communities.progress.com/pcom/docs/DOC-11318

OpenEdge documentation
http://communities.progress.com/pcom/docs/DOC-16074

Education Services

Technical Support

See also
The Progress Developer Studio for OpenEdge Guide on page 54
Help display options on page 54
Searching on page 55
Context sensitive help on page 56
Context sensitive help for NET controls Windows only on page 56
Visual Designer tutorials on page 57
Cheat sheets on page 57
Demos on page 57
Eclipse information resources

**Eclipse information resources**

Primary documentation for the Eclipse platform is available in the Eclipse Workbench User Guide, which is included in the online help for Progress Developer Studio for OpenEdge (Help > Help Contents). For newsgroups, documentation, and articles about Eclipse, go to the eclipse.org Web site.

**See also**

- The Progress Developer Studio for OpenEdge Guide on page 54
- Help display options on page 54
- Searching on page 55
- Context sensitive help on page 56
- Context sensitive help for NET controls Windows only on page 56
- Visual Designer tutorials on page 57
- Cheat sheets on page 57
- Demos on page 57
- OpenEdge information resources on page 58

**Tasks**

**Starting Progress Developer Studio for OpenEdge**

![Demo](image)

Creating Your First OpenEdge Architect Workspace and Project

**Note:** The OpenEdge Release 10.2B demos have not been updated for OpenEdge Release 11.0. Since 10.2B, OpenEdge Architect has been renamed as Progress Developer Studio for OpenEdge. In addition, OpenEdge Release 11.0 includes some feature enhancements and other changes. Despite the differences between 10.2B and 11.0, the demos are still a useful introduction to Progress Developer Studio for OpenEdge.

Start Progress Developer Studio for OpenEdge by running the Eclipse executable (OpenEdge-install-dir/DeveloperStudio3.6/eclipse/eclipse.exe) with a startup parameter that points to the appropriate Java Runtime Environment (JRE). The default is -vm OpenEdge-install-dir/DeveloperStudio3.6/jre/bin/javaw.exe).

In Windows, you can start Progress Developer Studio for OpenEdge from the Start menu by selecting one of the Developer Studio items under the Progress node.

**Note:** Some of the Developer Studio items include the Clean (-clean) option. The -clean option in the startup command clears any cached data before starting the Eclipse executable. This is useful when restarting Eclipse after modifying any plugins. It ensures that you see the latest version of all files.

In general, the Progress Developer Studio for OpenEdge startup sequence is:
1. Eclipse starts, modified by any startup parameters set on a command line or in icon properties. (See Setting Eclipse startup preferences.)

2. The **Workspace Launcher** dialog appears and prompts you to specify the path of a workspace. (See Selecting a workspace.)

3. The Eclipse Workbench appears in one of the following modes:

4. If you specified a new workspace, Eclipse opens displaying the Welcome to Progress Developer Studio page.

5. If you specified an existing workspace, Eclipse attempts to start the AVM and any required database connections. The Workbench displays an existing workspace in the state in which it was left the last time it was opened.

**See also**
Setting Eclipse startup preferences on page 60  
Selecting a workspace on page 60  
Setting up a shared AVM on page 62  
Setting workspace preferences on page 61

### Setting Eclipse startup preferences

- When you start Progress Developer Studio for OpenEdge from a desktop icon or from a Start menu selection, the default Eclipse startup command has the following format:

  OpenEdge-install-dir/DeveloperStudio3.6/eclipse/eclipse.exe -vm
  OpenEdge_install_dir/DeveloperStudio3.6/jre/bin/javaw.exe

- The default startup command points to the Eclipse executable (eclipse.exe) and the Java Runtime Environment (javaw.exe) that support Progress Developer Studio for OpenEdge.

- You might need to add startup parameters to the default so that Progress Developer Studio for OpenEdge runs on your system. For example, the default memory allocation (256MB) for Java might be inadequate in your operating environment. You can add the following command line parameters to increase the available memory for Java:

  -vmargs -Xmx<memory size>

- In Windows, you can open the properties of a desktop icon or a Start menu selection and add startup parameters in the Target field of the **Properties** dialog.

### Selecting a workspace

- When you start Progress Developer Studio for OpenEdge, Eclipse prompts you to specify a workspace in the **Workspace Launcher** dialog. A workspace is a folder that contains the content of your projects along with any project metadata.

- You can specify an existing workspace, or you can specify a folder in your file system that will function as a workspace. By default, the **Workspace Launcher** dialog maintains a list of the last five workspaces that you used.

- You can change workspaces when Progress Developer Studio for OpenEdge is running by selecting **File > Switch Workspace** from the main menu bar. Progress Developer Studio restarts when you change workspaces. If you want to work in multiple workspaces, you must start a separate instance of Progress Developer Studio for each workspace.
Setting workspace preferences

Workspace preferences are settings that apply to all the projects in a workspace. To set workspace preferences for OpenEdge projects:

1. Select Window > Preferences.
   Preference pages contributed by plugins are listed in the tree view in the left pane.
2. Expand the OpenEdge node.
3. Select a preference page and specify the desired options.

See also
Preferences and properties on page 51
Eclipse terminology on page 50
Setting the default AVM startup parameters on page 61
Setting up a shared AVM on page 62
OpenEdge workspace preferences on page 65

Setting the default AVM startup parameters

To set the default AVM startup parameters:

1. Select Window > Preferences.
   Preference pages contributed by plugins are listed in the tree view in the left pane.
2. Expand the Progress OpenEdge node.
3. Select the Startup node.
4. Specify the desired port ranges and startup parameters.
   Do not specify database connections as startup parameters. Use the Database Connections tab in the Project Properties dialog instead.

The startup parameters in the Default Development Startup field can be included, appended to, or overridden on a project's OpenEdge properties page. The default startup parameters can also be included, appended to, or overridden on the Shared OpenEdge AVM preference page.

See also
The ABL Virtual Machine (AVM) on page 52
Starting Progress Developer Studio for OpenEdge on page 59
Startup preference page on page 66
Shared AVM preference page on page 68
Setting up a shared AVM

To configure a shared AVM:

The shared AVM starts if there is a project associated with it.

1. Select Window > Preferences from the main menu bar of Progress Developer Studio for OpenEdge.
2. Open Progress OpenEdge > Shared OpenEdge AVM in the tree view of the Preferences dialog.
3. Set the AVM startup parameters on the Shared OpenEdge AVM preference page.
4. In the child nodes of the Shared OpenEdge AVM page, you can set the other properties (AppBuilder, Assemblies, Database connection, and PROPATH) that affect all the projects that use the shared AVM. In addition, you can choose which projects use the shared AVM.

See also

The ABL Virtual Machine (AVM) on page 52
Starting Progress Developer Studio for OpenEdge on page 59
Shared AVM preference page on page 68
Startup preference page on page 66

Working with OpenEdge perspectives

<table>
<thead>
<tr>
<th>Demos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using and Customizing Perspectives</td>
</tr>
<tr>
<td>Using and Managing Views</td>
</tr>
</tbody>
</table>

Note: The OpenEdge Release 10.2B demos have not been updated for OpenEdge Release 11.0. Since 10.2B, OpenEdge Architect has been renamed as Progress Developer Studio for OpenEdge. In addition, OpenEdge Release 11.0 includes some feature enhancements and other changes. Despite the differences between 10.2B and 11.0, the demos are still a useful introduction to Progress Developer Studio for OpenEdge.

The perspectives included with Progress Developer Studio for OpenEdge are OpenEdge AppBuilder, OpenEdge Server, Debugger, DB Navigator, OpenEdge Editor, OpenEdge Tools for Business Logic, and OpenEdge Visual Designer.

See also

Opening a perspective on page 63
Modifying a perspective on page 63
Setting a perspective to its default state on page 64
Eclipse terminology on page 50
OpenEdge perspectives and views on page 77
Opening a perspective

Opening a perspective

Demos
Using and Customizing Perspectives
Using and Managing Views

Note: The OpenEdge Release 10.2B demos have not been updated for OpenEdge Release 11.0. Since 10.2B, OpenEdge Architect has been renamed as Progress Developer Studio for OpenEdge. In addition, OpenEdge Release 11.0 includes some feature enhancements and other changes. Despite the differences between 10.2B and 11.0, the demos are still a useful introduction to Progress Developer Studio for OpenEdge.

To open an OpenEdge perspective in the Eclipse Workbench:

1. Select **Window > Open Perspective**.
2. Select an OpenEdge perspective from the list.

See also
- Eclipse terminology on page 50
- Modifying a perspective on page 63
- Setting a perspective to its default state on page 64
- OpenEdge perspectives and views on page 77

Modifying a perspective

Demos
Using and Customizing Perspectives
Using and Managing Views

Note: The OpenEdge Release 10.2B demos have not been updated for OpenEdge Release 11.0. Since 10.2B, OpenEdge Architect has been renamed as Progress Developer Studio for OpenEdge. In addition, OpenEdge Release 11.0 includes some feature enhancements and other changes. Despite the differences between 10.2B and 11.0, the demos are still a useful introduction to Progress Developer Studio for OpenEdge.

You can add views to a perspective, and you can close currently open views, thereby removing them from the perspective. You can save a given configuration of views as a custom perspective.

To modify the current perspective:
1. Select **Window > Show View** from the main menu bar.

2. Select Other to display a full set of views, organized in categories shown as folders, from which you can choose. You can select any view. Progress Developer Studio views are in folders whose names begin with "Progress."

**Note:** The views in the Progress OpenEdge Support Views folder are not intended to be selected. OpenEdge automatically displays these views when they are required. If selected from the Show View list, they may appear empty and non-functional.

3. To remove a view from the current perspective, close it by clicking the X on the view's tab.

4. To save your current configuration of views as a perspective, choose **Save Perspective As** from the **Window** menu and enter a name. You can overwrite an existing perspective by entering its name (a confirmation prompt appears), but it is not recommended. It is advisable to leave the standard Progress Developer Studio perspectives unchanged so that you retain the ability to return to a default state.

**See also**

- Eclipse terminology on page 50
- Opening a perspective on page 63
- Setting a perspective to its default state on page 64
- OpenEdge perspectives and views on page 77

### Setting a perspective to its default state

<table>
<thead>
<tr>
<th>Demos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using and Customizing Perspectives</td>
</tr>
<tr>
<td>Using and Managing Views</td>
</tr>
</tbody>
</table>

**Note:** The OpenEdge Release 10.2B demos have not been updated for OpenEdge Release 11.0. Since 10.2B, OpenEdge Architect has been renamed as Progress Developer Studio for OpenEdge. In addition, OpenEdge Release 11.0 includes some feature enhancements and other changes. Despite the differences between 10.2B and 11.0, the demos are still a useful introduction to Progress Developer Studio for OpenEdge.

1. After you change a standard perspective by adding or removing views, you might want to reset it to its unmodified state. To do so, choose **Window > Reset Perspective**, and click OK at the confirmation prompt.

2. The perspective returns to its last-saved state. To ensure that you can always return to the original, as-installed state, refrain from saving changes to a standard perspective. Instead, save your custom perspective with a new name.

**See also**

- Eclipse terminology on page 50
- Opening a perspective on page 63
- Modifying a perspective on page 63
- OpenEdge perspectives and views on page 77
Reference

OpenEdge workspace preferences

Note: The OpenEdge Release 10.2B demos have not been updated for OpenEdge Release 11.0. Since 10.2B, OpenEdge Architect has been renamed as Progress Developer Studio for OpenEdge. In addition, OpenEdge Release 11.0 includes some feature enhancements and other changes. Despite the differences between 10.2B and 11.0, the demos are still a useful introduction to Progress Developer Studio for OpenEdge.

Preferences usually apply either to the Eclipse framework or to individual plug-ins. To access preference dialogs, select Window > Preferences on the main menu bar. From the Preferences dialog, you can access pages containing general Eclipse settings as well as pages for individual plug-ins. For example, there are preferences that apply to all the editors that are installed in the Eclipse workbench, and there are also preferences that apply only to specific editors (such as the ABL Editor, the Visual Designer, and XML editors).

See also
- Eclipse terminology on page 50
- Preferences and properties on page 51
- Setting workspace preferences on page 61
- Views preference page on page 65
- Database Connections preferences on page 67
- Shared AVM preference page on page 68
- Preference pages for OpenEdge tools on page 70

Views preference page

The Views preference page allows you to set some general preferences that affect the Console, Outline, and Resources views. It also allows you to suppress or enable warning messages regarding running code that contains errors.

Select Window > Preferences to open the Preferences dialog. In the tree view, expand Progress OpenEdge and select Views to open the Views preference page.

The following options are available on the Views preference page:

| Show compile log in Console view | Display build messages in the standard Eclipse Console view. No build messages are displayed when this check box is cleared. |
| Show informational messages in Console view | Display informational messages (such as project startup status) in the standard Eclipse Console view. No informational messages are displayed when this check box is cleared. |
Show warning messages in Console view

Display warnings (such as the failure to start the AVM) in the standard Eclipse Console view. No warnings are displayed when this check box is cleared.

Split functions and procedures in Outline view

Display functions and procedures in separate categories in the Outline view while editing an ABL source file. The alternative is to show procedures and functions grouped together in a single category labeled Code Sections. Regardless of the setting here, you can toggle between these two display options by double-clicking the root node for one of the categories (Functions, Procedures, or Code Sections).

Show procedure libraries in Resources view

Include a fully expandable node in the Resources view that shows all procedure libraries available to each project. Clear this check box if you do not want this node to appear in the Resources tree.

Show warning message when running procedures that contain errors

Enable a pop-up alert that appears when you run a procedure with errors, allowing you to abort the Run command. The alert includes a Do not show this warning again option. (You can re-enable warnings after suppressing them by selecting this option again.)

See also

- Eclipse terminology on page 50
- Database connection profiles on page 52
- Working with perspectives on page 62
- OpenEdge perspectives and views on page 77
- OpenEdge workspace preferences on page 65

Startup preference page

The Startup preference page displays the default OpenEdge ABL Virtual Machine (AVM) startup parameters and the port number range.

Select Window > Preferences to open the Preferences dialog. In the tree view, expand Progress OpenEdge and select Startup to open the Startup preference page.

The following options are available on the Startup preference page:

| Lower Port Number, Upper Port Number | Specifies the range of ports on which Progress Developer Studio for OpenEdge can establish a connection with Eclipse. One port is used for each concurrent Eclipse session in which one or more OpenEdge tools are run. No changes to this port range should be necessary. |

Chapter 1: Introducing Progress Developer Studio for OpenEdge
Specifies any appropriate parameters for running the OpenEdge AVM.

Refer to OpenEdge Deployment: Startup Command and Parameter Reference for detailed information on startup parameters. You can find the manual in the Product Documentation section of the Progress Software Developer's Network Web site.

You can override or append to these parameters on a per-project basis on the OpenEdge properties page (Project>Properties>OpenEdge).

You can also override or append to these parameters on the Shared AVM preferences page (Window>Preferences>Shared OpenEdge AVM).

**Note:** Do not specify database connections as startup parameters. Use the Database Connections page for this purpose.

You can use a parameter file by including -pf parameter_file_path. Use a full path statement to avoid problems if multiple projects use the default parameters. A relative path statement is interpreted relative to the project working directory, which is usually different for each project.

### Default servers

Allows you to select an AppServer, OE Web Server, and PAS for OpenEdge as the default server.

### AppServer and OE Web Server

Specifies the default AppServer and OE Web Server for OpenEdge.

### Start AppServer and OE Web Server automatically

Starts the default AppServer and OE Web Server when you start Progress Developer Studio for OpenEdge.

### Progress Application Server for OpenEdge

Specifies the default Progress Application Server for OpenEdge.

### Start Progress Application Server automatically

Starts the default Progress Application Server for OpenEdge when you start Progress Developer Studio for OpenEdge.

### Database Connections preferences

The Database Connections preference page is used to maintain connections available to the project. The connections displayed in this page contain ABL connection information and optional SQL connection information.

This page provides a list of available database connections as defined for the workspace.

The following options are available on the Database Connections preference page:

| New               | Adds a new connection profile where you can define a new workspace database connection. For more information, see Add/Edit Connection Profile wizard. |

See also

- Setting workspace preferences on page 61
- OpenEdge workspace preferences on page 65
- Database connections preferences on page 67
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit</td>
<td>Modifies aspects of an existing connection profile. This button is disabled until a connection is selected. For more information, see Add/Edit Connection Profile wizard.</td>
</tr>
<tr>
<td>Copy</td>
<td>Copies an existing connection profile.</td>
</tr>
<tr>
<td>Remove</td>
<td>Deletes a connection profile.</td>
</tr>
<tr>
<td>Import</td>
<td>Imports a connection profile. When using this function, you must specify the location of the export file. You can import the entire file or individual resources. For information on importing database connections, see Importing database connections.</td>
</tr>
<tr>
<td>Export</td>
<td>Exports a connection profile. This function allows you to export either all connection profiles or individual database connection profiles to a system file. For information on exporting database connections, see Exporting database connections.</td>
</tr>
<tr>
<td>Import DF</td>
<td>(Progress OpenEdge databases only) Imports the data definitions from a .df file into the specified database.</td>
</tr>
<tr>
<td>Export DF</td>
<td>(Progress OpenEdge databases only) Exports the data definitions to a .df file for the specified database.</td>
</tr>
</tbody>
</table>

**See also**

OpenEdge database connection wizard on page 71

---

**Shared AVM preference page**

The Shared AVM preference page allows you to set some general options for the shared AVM that runs in the current workspace.

Select **Window > Preferences** to open the **Preferences** dialog. In the tree view, expand Progress OpenEdge and select Shared AVM to open the Shared AVM preference page.

The following options are available on the Shared AVM preference page:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Working directory</td>
<td>The directory where shared AVM configuration files (.propath, .dbconnection, assemblies.xml and toolbox.xml) are stored. By default, this is a directory named .sharedavm, which is in the workspace folder.</td>
</tr>
<tr>
<td>Temp directory</td>
<td>The directory in which temporary files are created. By default, this is the working directory specified when Progress Developer Studio for OpenEdge is installed.</td>
</tr>
</tbody>
</table>
Any appropriate parameters for starting the shared AVM. Refer to the ABL parameters section of *OpenEdge Deployment: Startup Command and Parameter Reference* for a list of AVM (also called the ABL client) startup parameters. You can find the manual in the *OpenEdge Product Documentation* section of the Progress Communities Web site.

**Note:** Do not specify database connections as startup parameters. Use the Database Connections page for this purpose.

You can use a parameter file by including `-pf parameter_file_path`. A relative path statement is interpreted relative to the project working directory.

If you check the Add default parameters option, the parameters in the Startup parameters box are appended to the defaults.

### Add default parameters

Use the default startup parameters.

**Note:** Although you can see the default startup parameters on this page, you cannot change them. Go to the Progress OpenEdge Startup preference page to change these defaults.

### Use TTY for runtime

Run ABL procedures in a character client window rather than a GUI window. The client window opens when you open the project.

### Use global Toolbox for Visual Designer

Use the global toolbox. The path of the global toolbox.xml file is specified on the Visual Designer preference page.

### Import Working Directory as Project

Import the shared AVM working directory as an Eclipse project, which makes it visible in the Resources view. If you do not import the shared AVM working directory, you must use some other file manager application (Windows Explorer, for example) to view the folder.

**Note:** An imported shared AVM working directory is an Eclipse project, and not an OpenEdge project. Therefore it does not have the special preferences and properties associated with an OpenEdge project.

**Note:** Shared AVM settings can be imported and exported from the **File** menu. In the Import or Export wizard, expand the Progress OpenEdge node and select Shared AVM Settings. The wizard allows you to specify a source or destination for the settings.

### See also

- [Preferences and properties](#)
- [The ABL Virtual Machine AVM](#)
- [Database connection profiles](#)
- [Setting workspace preferences](#)
- [Shared AVM properties](#)
Shared AVM properties

Under the Shared AVM preference page, there are property pages for AppBuilder, Assemblies, Database Connections, Projects, and PROPATH. You use these pages to set the relevant properties only for projects that use the shared AVM.

If a project does not use the shared AVM, use the project property pages Project > Properties > OpenEdge (from the main menu bar) to set these properties.

See also
Projects page on page 70

Projects page

The Projects page, found under the Shared OpenEdge AVM preference page, contains a list of the projects contained in the current workspace. Projects that use the shared AVM are indicated by a check mark. You can add or remove the check mark to change shared AVM usage.

Note: If you change an existing project to use the shared AVM, you may need to update the shared AVM settings to include the PROPATH, assemblies, and database connections of the project.

See also
The ABL Virtual Machine AVM on page 52
Setting workspace preferences on page 61
OpenEdge workspace preferences on page 65
Shared AVM preferences page on page 68

Preference pages for OpenEdge tools

The following topics contain information on the preference pages for various OpenEdge tools:
"Server" in the AppServer online help
"Class Cache" in the ABL Editor online help
"Customization" in the Customization online help
"Launching" in Running and Debugging ABL Programs online help
"Debug" in Running and Debugging ABL Programs online help
"Editor" in the ABL Editor online help
"Meta Catalog" in the Meta Catalog online help
"Tools for Business Logic" in the Tools for Business Logic online help
"Visual Designer" in the Visual Designer online help
"Parsing" in the ABL Editor online help
"AppBuilder" in the GUI Desigener online help
Views in the Getting Started online help
Customer Experience Improvement Program preferences

The Customer Experience Improvement Program preference enables Progress Software to collect data related to product usage trends and patterns. This data helps us improve the product. It collects only software and hardware configuration of your system and how you use Progress Software Developer for OpenEdge.

Select Window > Preferences to open the Preferences dialog box. In the tree view, expand Progress OpenEdge > Advanced > Customer Experience Improvement Program. This dialog box also opens when you open the workspace for the first time.

Select the Sign up for Customer Experience Improvement Program option to enable this option.

Note: If you are using a classroom edition of OpenEdge, this option is enabled by default and the Customer Experience Improvement Program preference page is not visible to you.

OpenEdge database connection wizard

Demos

Defining Workspace Database Connections

Note: The OpenEdge Release 10.2B demos have not been updated for OpenEdge Release 11.0. Since 10.2B, OpenEdge Architect has been renamed as Progress Developer Studio for OpenEdge. In addition, OpenEdge Release 11.0 includes some feature enhancements and other changes. Despite the differences between 10.2B and 11.0, the demos are still a useful introduction to Progress Developer Studio for OpenEdge.

The OpenEdge database connection wizard allows you to create database connection profiles that are available to any project in a workspace. You can access the wizard by selecting New or Edit from the Database Connections preferences on page 67 dialog.

This section describes each page in the OpenEdge database connection wizard.
Add/Edit OpenEdge database connection profile wizard

| **Demo** | Defining Workspace Database Connections |

**Note:** The OpenEdge Release 10.2B demos have not been updated for OpenEdge Release 11.0. Since 10.2B, OpenEdge Architect has been renamed as Progress Developer Studio for OpenEdge. In addition, OpenEdge Release 11.0 includes some feature enhancements and other changes. Despite the differences between 10.2B and 11.0, the demos are still a useful introduction to Progress Developer Studio for OpenEdge.

You can access the Connection Profile wizard by selecting New or Edit from the Database Connections preference page. T

**Note:** The wizard is labeled Add Connection Profile when you select New and Edit Connection Profile when you select Edit. However, both pages contain identical fields and controls.

The Add/Edit OpenEdge Database Connection page contains the following options:

<table>
<thead>
<tr>
<th><strong>Connection name</strong></th>
<th>This is a unique name that describes the connection. This is a required field.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Physical Name</strong></td>
<td>The complete path name of an OpenEdge database. You can specify the path relative to the current workspace folder.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Describes the type of database connection. This is an optional field.</td>
</tr>
<tr>
<td><strong>Logical Name</strong></td>
<td>The logical database name.</td>
</tr>
</tbody>
</table>
| **Host name**       | The host name must be **localhost** when you are using the OpenEdge Personal RDBMS database (which is the default RDBMS packaged with Progress Developer Studio for OpenEdge). You cannot connect to an OpenEdge Personal RDBMS database that is running remotely.  
If you are connecting to an OpenEdge Workgroup or to an OpenEdge Enterprise RDBMS, you can specify the host name of the system where the database server process is running. If the database is running on the local system, you can use **localhost** in place of the host name. |
| **Service/port**    | Identifies the service or the port number to use on the host machine when connecting to a broker process. |

**Note:** If you are connecting to an OpenEdge database, you can find appropriate values by checking the configuration settings for the database in Progress Explorer.

| **User ID** | If the user ID and the password were previously created, enter the information in the User ID field. You must specify a user ID. |
The password to access the database.

<table>
<thead>
<tr>
<th>Password</th>
<th>The password to access the database.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>Specifies a list of aliases that are created at the start of the client session. The validation is the same as the logical name.</td>
</tr>
<tr>
<td>Group</td>
<td>Used for organizing your connection profiles. You can add a new group or you can select an existing group.</td>
</tr>
<tr>
<td>Other Parameters</td>
<td>Displays a string of supported database connection parameters.</td>
</tr>
</tbody>
</table>

**See also**

- Database connection profiles on page 52

**Define a SQL connection profile**

<table>
<thead>
<tr>
<th>Demo</th>
<th>Defining Workspace Database Connections</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>The OpenEdge Release 10.2B demos have not been updated for OpenEdge Release 11.0. Since 10.2B, OpenEdge Architect has been renamed as Progress Developer Studio for OpenEdge. In addition, OpenEdge Release 11.0 includes some feature enhancements and other changes. Despite the differences between 10.2B and 11.0, the demos are still a useful introduction to Progress Developer Studio for OpenEdge.</td>
</tr>
</tbody>
</table>

When you create a new database connection using the connection profile wizard, you can specify whether or not to define a SQL connection. A SQL connection is necessary to display the tables and columns in the DB Structure View.

The following options are available on the Define a SQL connection profile page:

<table>
<thead>
<tr>
<th>Define SQL connection</th>
<th>When this checkbox is selected, you can define a new SQL connection, or you can assign a connection that already exists.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add new SQL connection</td>
<td>The Add new SQL connection radio button indicates that the connection profile uses a new SQL connection. Select this option, then click Next to display the Add New SQL Connection Profile page.</td>
</tr>
<tr>
<td>Use existing SQL connection</td>
<td>The Use existing SQL connection radio button indicates that the connection profile uses a previously defined SQL connection. Existing connections are listed. To assign an existing connection, select the checkbox nearest the connection profile and click Next. The next page in the connection profile wizard (Define Database Server configuration) appears.</td>
</tr>
</tbody>
</table>

**Note:** When you select the Use existing SQL connection radio button, items in the browser become selectable.
See OpenEdge Data Management: SQL Development and OpenEdge Data Management: SQL Reference manuals for more information on SQL connections. You can find OpenEdge manuals in the Product Documentation section of the Progress Software Developer’s Network Web site.

## Add a SQL connection profile

When you create a new database connection using the connection profile wizard, you are prompted to create a new SQL connection, or assign an existing connection to the database. The Add SQL connection profile page allows you to define a new SQL connection. This page appears when you select the Define SQL Connection checkbox and the corresponding Add new SQL connection radio button, then click Next.

**Note:** By default, the Define SQL Connection checkbox and the corresponding Add new SQL connection radio button are enabled.

The following options are available on the Add SQL connection profile page:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection name</td>
<td>This is a unique name that describes the connection. This field is automatically populated if you defined the connection name in the first page of the database connection wizard.</td>
</tr>
<tr>
<td>Driver</td>
<td>Specifies the JDBC driver for the connection profile. Select the driver that is appropriate for the database from the drop-down menu. For more information, see Configuring a JDBC driver.</td>
</tr>
<tr>
<td>User ID</td>
<td>Specifies the User ID for the SQL connection profile. This field is automatically populated if you defined the user ID in the first page of the database connection wizard. You can omit both the user ID and the optional password from the connection profile if the auto-login checkbox is not selected. However, you will be prompted for the user ID and password every time you make a connection.</td>
</tr>
<tr>
<td>Password</td>
<td>Specifies the password to access the database.</td>
</tr>
<tr>
<td>Open on Eclipse startup</td>
<td>Opens the database connection when Progress Developer Studio for OpenEdge starts.</td>
</tr>
</tbody>
</table>

**Note:** If you enable this option, you still must enter a valid user ID and an optional password. This option slows the startup of Eclipse. If you select the Auto-start a database server on the next page of database connection wizard, you cannot use this option; the system will attempt to connect before the database automatically starts.
<table>
<thead>
<tr>
<th><strong>Auto-Login on connect</strong></th>
<th>Automatically logs into the database when OpenEdge Architect starts. The system attempts to log in using the provided user ID and password.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>URL details</strong></td>
<td>Allows you to edit URL information for your system. Selecting this checkbox enables all URL fields.</td>
</tr>
<tr>
<td><strong>Host name</strong></td>
<td>The host name must be <strong>localhost</strong> when you are using the OpenEdge Personal RDBMS database (which is the default RDBMS packaged with Progress Developer Studio for OpenEdge). You cannot connect to an OpenEdge Personal RDBMS database that is running remotely. This value defaults to the host name entered on the first page. If you are connecting to an OpenEdge Workgroup or OpenEdge Enterprise RDBMS, you can specify the host name of the system where the database server process is running. If the database is running on the local system, you can use <strong>localhost</strong> in place of the host name.</td>
</tr>
<tr>
<td><strong>Service/port</strong></td>
<td>Identifies the service or the port number to use on the host machine when connecting to a broker process.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> If you are connecting to an OpenEdge database, you can find appropriate values by checking the configuration settings for the database in Progress Explorer.</td>
</tr>
<tr>
<td><strong>Database name</strong></td>
<td>Represents the name of the database.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> If you are connecting to an OpenEdge database, check for the correct name in the <strong>Databases</strong> folder in Progress Explorer.</td>
</tr>
<tr>
<td><strong>SQL parameters</strong></td>
<td>Displays any additional SQL parameters delimited by a semi-colon in the format key1=value; key2=value.</td>
</tr>
</tbody>
</table>

See OpenEdge Data Management: SQL Development and OpenEdge Data Management: SQL Reference manuals for more information on SQL connections. You can find OpenEdge manuals in the **Product Documentation** section of the Progress Software Developer’s Network Web site.

**See also**

- Database connection profiles on page 52
- Preference pages for OpenEdge tools on page 67
Define an OpenEdge database server configuration

Note: The OpenEdge Release 10.2B demos have not been updated for OpenEdge Release 11.0. Since 10.2B, OpenEdge Architect has been renamed as Progress Developer Studio for OpenEdge. In addition, OpenEdge Release 11.0 includes some feature enhancements and other changes. Despite the differences between 10.2B and 11.0, the demos are still a useful introduction to Progress Developer Studio for OpenEdge.

This page appears when you define the database server configuration when Progress Developer Studio for OpenEdge starts. The fields in this dialog apply databases configured locally; use Progress Explorer to configure remote databases.

Note: The information in this dialog is optional. Use Progress Explorer to configure remote databases.

The following options are available on the Define an OpenEdge database server configuration page:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto-start database server</td>
<td>Automatically starts the database server when you start Progress Developer Studio for OpenEdge. Use this option only if you are starting a local database.</td>
</tr>
<tr>
<td>Physical name</td>
<td>Specifies the pathname of the OpenEdge database. This field is automatically populated if you have defined the connection name in the first page of the database connection wizard.</td>
</tr>
<tr>
<td>Service/Port</td>
<td>Identifies the service or port number to use on the host machine when connecting to a broker process. This field is automatically populated if you defined the connection name in the first page of the database connection wizard.</td>
</tr>
<tr>
<td>Parameters</td>
<td>Specifies a string of supported database connection parameters.</td>
</tr>
<tr>
<td>Auto-shutdown database server</td>
<td>Automatically stops the database server when you quit Progress Developer Studio for OpenEdge.</td>
</tr>
</tbody>
</table>
OpenEdge perspectives and views

Demos
Using and Customizing Perspectives
Using and Managing Views

Note: The OpenEdge Release 10.2B demos have not been updated for OpenEdge Release 11.0. Since 10.2B, OpenEdge Architect has been renamed as Progress Developer Studio for OpenEdge. In addition, OpenEdge Release 11.0 includes some feature enhancements and other changes. Despite the differences between 10.2B and 11.0, the demos are still a useful introduction to Progress Developer Studio for OpenEdge.

The topics in this section summarize the perspectives and views that are available with Progress Developer Studio for OpenEdge:

For more detailed information about using these perspectives and views, refer to the appropriate online help volumes.

See also
The Eclipse platform on page 50
Opening a perspective on page 63
Modifying a perspective on page 63
Setting a perspective to its default state on page 64

OpenEdge default perspectives

Demos
Using and Customizing Perspectives
Using and Managing Views

Note: The OpenEdge Release 10.2B demos have not been updated for OpenEdge Release 11.0. Since 10.2B, OpenEdge Architect has been renamed as Progress Developer Studio for OpenEdge. In addition, OpenEdge Release 11.0 includes some feature enhancements and other changes. Despite the differences between 10.2B and 11.0, the demos are still a useful introduction to Progress Developer Studio for OpenEdge.

OpenEdge perspectives are available when you select Window > Open Perspective from the main menu bar of Progress Developer Studio for OpenEdge.

The following table lists the OpenEdge perspectives and the default views that each perspective includes:
<table>
<thead>
<tr>
<th>Perspective</th>
<th>Default views</th>
<th>Associated help volume in the Progress Developer Studio for OpenEdge</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenEdge Editor</td>
<td>• Resources</td>
<td>ABL Editor</td>
</tr>
<tr>
<td></td>
<td>• Outline</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• DB Structure</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Console</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Problems</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Tasks</td>
<td></td>
</tr>
<tr>
<td>OpenEdge Visual Designer (Windows only)</td>
<td>• Resources</td>
<td>Visual Designer</td>
</tr>
<tr>
<td></td>
<td>• Properties</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Outline</td>
<td></td>
</tr>
<tr>
<td>OpenEdge Debugger</td>
<td>• Debug</td>
<td>Debugger</td>
</tr>
<tr>
<td></td>
<td>• Variables</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Breakpoints</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Expressions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Dynamic Objects</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Console</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Tasks</td>
<td></td>
</tr>
<tr>
<td>OpenEdge AppBuilder</td>
<td>• AppBuilder Framework</td>
<td>GUI Designer</td>
</tr>
<tr>
<td></td>
<td>• Resources</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Outline</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• DB Structure</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Console</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Problems</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Tasks</td>
<td></td>
</tr>
<tr>
<td>Perspective</td>
<td>Default views</td>
<td>Associated help volume in the Progress Developer Studio for OpenEdge: Online Help</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>------------------------------------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| OpenEdge Server                   | • Resources  
• Outline  
• DB Structure  
• Properties  
• Console  
• Problems  
• Tasks  
• Servers  
• OpenEdge Server Monitor        | AppServer  
WebSpeed                                                                 |
| OpenEdge Tools for Business Logic | • Resources  
• Properties  
• Outline  
• DB Structure  
• Connections  
• Problems                                                                 | Tools for Business Logic |
| Progress DB Navigator             | • Connection Info  
• DB Structure  
• Navigator  
• Connections  
• DB Details  
• SQL  
• Results  
• SQL History  
• SQL Editor                                                                 | Progress DB Navigator  
(outside of the Progress Developer Studio for OpenEdge Guide) |

**See also**
The Eclipse platform on page 50  
Opening a perspective on page 63  
Modifying a perspective on page 63  
Setting a perspective to its default state on page 64  
OpenEdge views on page 80
OpenEdge views

<table>
<thead>
<tr>
<th>Demo</th>
<th>Using and Managing Views</th>
</tr>
</thead>
</table>

**Note:** The OpenEdge Release 10.2B demos have not been updated for OpenEdge Release 11.0. Since 10.2B, OpenEdge Architect has been renamed as Progress Developer Studio for OpenEdge. In addition, OpenEdge Release 11.0 includes some feature enhancements and other changes. Despite the differences between 10.2B and 11.0, the demos are still a useful introduction to Progress Developer Studio for OpenEdge.

You can add a view to a perspective by selecting **Windows > Show View** menu from the Progress Developer Studio for OpenEdge main menu bar. Views are organized in categories in the Show View window that appears when you select **Window > Show View > Other**. Within each category, views are listed in alphabetical order.

**See also**

- The Eclipse platform on page 50
- Opening a perspective on page 63
- Modifying a perspective on page 63
- Setting a perspective to its default state on page 64
- OpenEdge default perspectives on page 77
Introducing OpenEdge Projects

Progress Developer Studio for OpenEdge, like other Eclipse-based development environments, is project based. Projects exist in the context of a workspace. Project folders contain the files and sub-folders that are the project's resources.

The features available for OpenEdge projects include:

• A predefined set of OpenEdge facets and project types.

• A set of commonly used project types for OpenEdge application development and a framework to create custom project types, as required. These project types are convenient configurations for a specific type of project that is provided by default in Progress Developer Studio for OpenEdge.

• A facet for each project type that is configured by default. For example, a WebSpeed project type has an OpenEdge WebSpeed facet, and so on. Progress Developer Studio for OpenEdge also creates a facet for each custom project type that you define.

With OpenEdge projects, you can create:

• All new OpenEdge projects as faceted projects.

• A specialized project type for the current project with the New OpenEdge Project wizard.

All OpenEdge projects that you create in Progress Developer Studio for OpenEdge are faceted projects. A faceted project uses one or more predefined configurations called facets. A facet is used as a way of adding project functionality that you can turn on or off. Within OpenEdge projects, you can use facets to add a specific module type to a project.

Note: A module is used to organize content that is being developed for a server and usually consists of a hierarchy of files that reside in the Eclipse workspace.
When you add a facet to a project, the facet is used as a marker for performing certain setup requirements or for marking the project as a specific project type.

You can use the **New OpenEdge Project** wizard or the **Project Properties** dialog to create a specialized project type for your current project. You can save this project type, but it will only be available within the workspace of the project.

**Migrating older versions of OpenEdge projects**

You can migrate an OpenEdge project created in a Progress Developer Studio for OpenEdge version prior to 11.0, to its current version and convert it to a faceted project. When you open the old project, by default, Progress Developer Studio for OpenEdge installs the OpenEdge project facet to it. It also updates the project runtime version and migrates the project properties and the Visual Designer properties.

**Note:** When you try to open an 11.0 version of the workspace in Progress Developer Studio for OpenEdge 10.2B, and the facets associated with the project are not available in 10.2B, a warning message is displayed on the Project Facets page in Project Properties.

When you convert a normal OpenEdge project to a faceted OpenEdge project, it automatically creates a `.setting` folder that contains the configuration files in the project. You must add this folder to the source control system.

For details, see the following topics:

- Concepts
- Tasks
- Reference

## Concepts

**Note:** For general information about projects in Eclipse, see *Working with Projects, Folders, and Files* (in the Eclipse Workbench User's Guide).

## Introducing OpenEdge Projects

Progress Developer Studio for OpenEdge, like other Eclipse-based development environments, is project based. Projects exist in the context of a workspace. Project folders contain the files and sub-folders that are the project's resources.

The features available for OpenEdge projects include:

- A predefined set of OpenEdge facets and project types.
- A set of commonly used project types for OpenEdge application development and a framework to create custom project types, as required. These project types are convenient configurations for a specific type of project that is provided by default in Progress Developer Studio for OpenEdge.
- A facet for each project type that is configured by default. For example, a WebSpeed project type has an OpenEdge WebSpeed facet, and so on. Progress Developer Studio for OpenEdge also creates a facet for each custom project type that you define.

With OpenEdge projects, you can create:
• All new OpenEdge projects as faceted projects.

• A specialized project type for the current project with the New OpenEdge Project wizard.

All OpenEdge projects that you create in Progress Developer Studio for OpenEdge are faceted projects. A faceted project uses one or more predefined configurations called facets. A facet is used as a way of adding project functionality that you can turn on or off. Within OpenEdge projects, you can use facets to add a specific module type to a project.

**Note:** A module is used to organize content that is being developed for a server and usually consists of a hierarchy of files that reside in the Eclipse workspace.

When you add a facet to a project, the facet is used as a marker for performing certain setup requirements or for marking the project as a specific project type.

You can use the **New OpenEdge Project** wizard or the **Project Properties** dialog to create a specialized project type for your current project. You can save this project type, but it will only be available within the workspace of the project.

**Migrating older versions of OpenEdge projects**

You can migrate an OpenEdge project created in a Progress Developer Studio for OpenEdge version prior to 11.0, to its current version and convert it to a faceted project. When you open the old project, by default, Progress Developer Studio for OpenEdge installs the OpenEdge project facet to it. It also updates the project runtime version and migrates the project properties and the Visual Designer properties.

**Note:** When you try to open an 11.0 version of the workspace in Progress Developer Studio for OpenEdge 10.2B, and the facets associated with the project are not available in 10.2B, a warning message is displayed on the Project Facets page in Project Properties.

When you convert a normal OpenEdge project to a faceted OpenEdge project, it automatically creates a .setting folder that contains the configuration files in the project. You must add this folder to the source control system.

**OpenEdge project and resource data files**

When you are working in an OpenEdge project, Progress Developer Studio for OpenEdge often creates data files that contain configuration information for the project or for project resources. You often include these data files when you are importing, exporting, or sharing projects or project resources.

Most project data files are visible in a project's **Resources** view. Some files (.resx, for example) are hidden by default. The **Resources** view has a **Filters** option on its drop-down display menu where you can choose which file types to display or hide.

**Note:** You should not edit project data files. You could introduce errors that would make the project or some of its resources unusable. Project data files are usually updated automatically in Progress Developer Studio for OpenEdge when you change settings using property or preferences pages. Some project data files are automatically updated when you add, modify, or delete resources.

Some of the OpenEdge project data files are:
### Table

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>assemblies.xml</td>
<td>A list of the assemblies where .NET classes are stored that is referenced at compile time and run time.</td>
</tr>
<tr>
<td>.dbconnection</td>
<td>A list of the database connections used by a project.</td>
</tr>
<tr>
<td>filename.resx</td>
<td>A .NET resource file that can contain localized values for controls, as well as binary graphic files.</td>
</tr>
<tr>
<td>filename.esboe</td>
<td>A list of the Sonic Enterprise Service Bus (ESB) annotations in ABL code that can be exported to a Sonic environment.</td>
</tr>
<tr>
<td>.project</td>
<td>General project information, such as the name and the nature of the project.</td>
</tr>
<tr>
<td>.propath</td>
<td>A list of the directories in the project's PROPATH.</td>
</tr>
<tr>
<td>toolbox.xml</td>
<td>A list of the changes to the Visual Designer Toolbox.</td>
</tr>
</tbody>
</table>

### See also
Progress OpenEdge project properties pages

### Configuration variables

You can use variables in place of explicit directory paths in certain fields in project property pages. Besides reducing typing errors, using variables and then sharing settings can lessen the need to specify new paths when importing properties to a project. The following predefined variables are available at the project and workspace level:

- `@{DLC}` - The root directory of your OpenEdge installation.
- `@{ROOT}` - The project root directory, specified when you create the project.
- `@{WORK}` - The project working directory, specified on the Progress OpenEdge properties page. (You can enter a period (.) to represent `{WORK}`.)
- `@{RCODE}` - The directory (if any) specified in the Build destination field on the Build and Source page.
- `@{APP}` - The application root directory, if any, specified on the Custom page.
- `@{DB}` - The database root directory, if any, specified on the Custom page.
- `@{TEMP}` - The directory where temporary files are created.

The following is the list of the locations where these variables apply:

- In the Configuration File field (for Dynamics) on the AppBuilder properties page.
- In the Build destination field and Source entry on the Build properties page.
- In the Source tab of the Build properties page.
In the alias directory entries on the SpeedScript properties page.

• In PROPATH entry in the PROPATH properties page.

• In Assemblies properties page for assemblies.xml file location

• In File Destinations properties page for ESB, BPM, and REST invocation files

In addition to the above predefined variables, you can also define new configuration variables and use them to define project properties. See Defining new configuration variables for more information.

See also
Creating a new OpenEdge project
Exporting and importing project resources and properties on page 100
Sharing project properties on page 100
Progress OpenEdge properties page on page 112
AppBuilder properties page Windows only on page 114
Build properties page on page 116
Custom properties page on page 119
Database Connections properties page on page 120
PROPATH properties page on page 122

Project type options for application development

In OpenEdge 11.7, the process of creating a project has been simplified. The New OpenEdge Project wizard displays three categories for front-end (client) and back-end development (server) and general application development. When you open the Create an OpenEdge Project page, you have to select from the following options. How these options map to previous version of the product is shown in the following table for your convenience:

<table>
<thead>
<tr>
<th>Option in previous versions</th>
<th>Option in 11.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenEdge</td>
<td>Do not select any category and click Finish or select OpenEdge Basic in the General category.</td>
</tr>
<tr>
<td>ABLUnit</td>
<td>Select ABLUnit in the General category.</td>
</tr>
<tr>
<td>Dynamics</td>
<td>Select Dynamics in the General category.</td>
</tr>
<tr>
<td>Custom</td>
<td>Select Custom in the General category.</td>
</tr>
<tr>
<td>AppServer</td>
<td>Select PAS for OpenEdge or Classic server type and APSV transport in the Server category.</td>
</tr>
<tr>
<td>ABL Web App project with WebSpeed (WebHandler) service</td>
<td>Select the PAS for OpenEdge server and WEB transport in the Server category.</td>
</tr>
<tr>
<td>ABL Web App project with REST (Mapped RPC) service</td>
<td>Select the PAS for OpenEdge server and REST transport in the Server category.</td>
</tr>
</tbody>
</table>
**Project types in Progress Developer Studio for OpenEdge**

Progress Developer Studio for OpenEdge provides project types for each specific development domain. This enables better visibility over the organization of code-base and editing the functionality based on the context. It also provides a generic framework to create custom project types of your choice. In a custom project type, you can specify the folder layout, default files in each folder, PROPATH settings for folders in each project type, and several other project properties.

The following table lists the project types that Progress Developer Studio for OpenEdge supports and the default content that each project type includes:

<table>
<thead>
<tr>
<th>Project type</th>
<th>Default runtime</th>
<th>Default PROPATH</th>
<th>Startup parameters</th>
<th>Default files</th>
<th>Default folder</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;custom&gt;</td>
<td>&lt;Runtime that you define&gt;</td>
<td>@{WORK},&lt;br/&gt;,@{default entries},@{entries that you define}&gt;</td>
<td>&lt;NONE&gt;</td>
<td>&lt;Files that you define&gt;</td>
<td>&lt;Folders that you define&gt;</td>
</tr>
<tr>
<td>AppServer</td>
<td>TT</td>
<td>@{ROOT}\AppServerContent, @{WORK},&lt;default entries&gt;</td>
<td>&lt;NONE&gt;</td>
<td>&lt;NONE&gt;</td>
<td>AppServerContent</td>
</tr>
<tr>
<td>Project type</td>
<td>Default runtime</td>
<td>Default PROPATH</td>
<td>Startup parameters</td>
<td>Default files</td>
<td>Default facet</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------</td>
<td>--------------------------------------------------------------------------------</td>
<td>--------------------</td>
<td>---------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>ChUI</td>
<td>TTY</td>
<td>@{WORK}, &lt;default entries&gt;</td>
<td>&lt;NONE&gt;</td>
<td>&lt;NONE&gt;</td>
<td>&lt;NONE&gt;</td>
</tr>
<tr>
<td>Dynamics (Windows only)</td>
<td>GUI</td>
<td>@{WORK}, @{DLC}\gui\dynamics, @{DLC}\src\dynamics, @{DLC}\src\dynamics\af\sup2, &lt;default entries&gt;</td>
<td>&lt;NONE&gt;</td>
<td>&lt;NONE&gt;</td>
<td>&lt;NONE&gt;</td>
</tr>
<tr>
<td>GUI for .NET (Windows only)</td>
<td>GUI</td>
<td>@{WORK}, &lt;default entries&gt;</td>
<td>-preloadCLR' assemblies.xml; Form1.cls; RunForm.p</td>
<td>&lt;NONE&gt;</td>
<td></td>
</tr>
<tr>
<td>OpenEdge</td>
<td>GUI</td>
<td>@{WORK}, &lt;default entries&gt;</td>
<td>&lt;NONE&gt;</td>
<td>&lt;NONE&gt;</td>
<td>&lt;NONE&gt;</td>
</tr>
<tr>
<td>WebSpeed</td>
<td>TTY</td>
<td>@{ROOT}\DynamicContent, @{WORK}, &lt;default entries&gt;</td>
<td>&lt;NONE&gt;</td>
<td>&lt;NONE&gt;</td>
<td>DynamicContent, StaticContent, SpeedScript</td>
</tr>
<tr>
<td>REST</td>
<td>TTY</td>
<td>@{ROOT}\AppServerContent, @{WORK}, &lt;default entries&gt;</td>
<td>NONE</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>ABL Web App</td>
<td>TTY</td>
<td>@{ROOT}\AppServer, @{ROOT}\OpenEdgeSrc, @{WORK}, &lt;default entries&gt;</td>
<td>NONE</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>WebUI</td>
<td></td>
<td>Project types in Progress Developer Studio for OpenEdge</td>
<td>NONE</td>
<td>NONE</td>
<td>NONE</td>
</tr>
</tbody>
</table>

Each project type is associated with one or more facets and each facet is associated with one or more pages displayed in the New OpenEdge Project wizard.

The following is a list of facets that Progress Developer Studio for OpenEdge provides:

- OpenEdge AppServer facet
- ChUI facet
Chapter 2: Introducing OpenEdge Projects

- Dynamic facet (Windows only)
- GUI for .NET facet (Windows only)
- OpenEdge facet
- Static Web facet (configuration of OpenEdge facet and WTP Static Web facet)
- OpenEdge WebSpeed facet
- REST
- Data Object
- ABL Service
- WebUI

See also
Creating a new OpenEdge project
New OpenEdge project wizard on page 105

Custom project type
A custom project type is a combination of different project types added to a single project. In a custom project type, you can specify the folder layout, the default files in each folder, and the PROPATH settings for folders in each project. You can also share these custom project types with other developers. The custom project types you create can be used as templates for new projects.

For information on creating custom projects, see Creating a custom project on page 93.

See also
Project types in Progress Developer Studio for OpenEdge on page 86
Creating a custom project on page 93
New OpenEdge Project wizard on page 105

AppServer project type
An AppServer project type is an OpenEdge project that is specialized for AppServer applications.

For information on AppServer and creating AppServer projects, see the AppServer help.

See also
Project types in Progress Developer Studio for OpenEdge on page 86
New OpenEdge Project wizard on page 105

ChUI project type
A ChUI project type is an OpenEdge project specialized for character-based UI development and uses TTY as the runtime.

For information on AppServer and creating ChUI projects, see Creating a ChUI project on page 91.

See also
Project types in Progress Developer Studio for OpenEdge on page 86
Dynamics project type (Windows only)

A Dynamics project type is an OpenEdge project specialized for developing applications incorporating Progress Dynamics. By default, a Dynamics project type uses GUI as the runtime and adds Dynamics PROPATH entries to the project. It allows you to choose the Dynamics configuration file and the session type on the Define Dynamics configuration page (Windows only) on page 110 page.

For information on creating a Dynamics project, see Creating a Dynamics project (Windows only) on page 94.

See also

Project types in Progress Developer Studio for OpenEdge on page 86
Creating a Dynamics project (Windows only) on page 94
New OpenEdge Project wizard on page 105

GUI for .NET project type (Windows only)

A GUI for .NET project type is an OpenEdge project specialized for .NET-based GUI development. By default, a GUI for .NET project type uses GUI as the runtime.

For information on creating a GUI for .NET project, see Creating a GUI for .NET project (Windows only) on page 92.

See also

Project types in Progress Developer Studio for OpenEdge on page 86
Creating a GUI for NET project (Windows only) on page 92
New OpenEdge Project wizard on page 105

OpenEdge project type

An OpenEdge project type is a standard OpenEdge project without specialized properties.

For information on creating an OpenEdge project, see Creating a standard OpenEdge project on page 92.

See also

Project types in Progress Developer Studio for OpenEdge on page 86
Creating a standard OpenEdge project on page 92
New OpenEdge Project wizard on page 105
WebSpeed project type

A WebSpeed project type is an OpenEdge project specialized for WebSpeed applications. It is characterized by an OpenEdge WebSpeed facet. A WebSpeed project has two kinds of resources: Static and Dynamic. Static files are published to a Web server and dynamic files are published to a WebSpeed server. Static files can be HTML files, CSS files, JS files, and images. They are served by an HTTP Web server directly. Dynamic files are CGI wrappers or embedded SpeedScript files. These files are served by a WebSpeed broker when the Web server delegates the HTTP request to it through a WebSpeed messenger.

**Note:** If you want to develop new WebSpeed applications that run on PAS for OpenEdge, create an ABL Web App project.

A WebSpeed project contains two modules:

- OpenEdge static Web content module
- OpenEdge WebSpeed dynamic content module

An OpenEdge WebSpeed dynamic content module comprises of a set of directories under the project folder that are published to the WebSpeed broker.

A WebSpeed project type uses TTY as the default runtime.

For information on WebSpeed and creating a WebSpeed project, see the WebSpeed help.

**See also**

- Project types in Progress Developer Studio for OpenEdge on page 86
- New OpenEdge Project wizard on page 105

Tasks

Creating a new OpenEdge project

Progress Developer Studio for OpenEdge provides various project types for OpenEdge application development and also a framework to create custom project types, as required. Facets for each project type is configured by default.

To create an OpenEdge project:

1. Select File > New > OpenEdge project from the Progress Developer Studio for OpenEdge main menu bar or the Project Explorer view context (right-click) menu. The New OpenEdge Project wizard opens and displays the Create an OpenEdge Project page.

2. Enter a name in the Project name field. The root folder for the new project with this name is created in your current workspace. To choose a different location for the project, clear the Use default check box. Then either browse to or enter the path to the required location.

3. Do one of the following:

   - To create a project for the back-end development of the application, select the Server category from the left section.
Select the server type that you want to deploy the services to and the transport type that you want to use for deploying the application.

- To create a project for the front-end development of the application, select the **Client** category from the left section.

  Select the **Desktop Application** or the **Web Application** option button to create a project for a desktop application or a web application respectively and select one of the project types.

- To create a basic, ABLUnit, Dynamics, or custom OpenEdge project, select the **General** category from the left section.

**Note:** If you do not select a category (Client, Server, or General) and click **Finish**, a basic OpenEdge project without any specialization is created.

4. Click **Finish**. The project is created and is displayed in the **Project Explorer** view.

5. To review or modify project properties after the project is created, right-click the project in the **Project Explorer** view and select **Properties**.

**Note:** For large applications with multiple modules, it is recommended that you create separate projects for the modules. You must also create separate projects for modules that have a different PROPATH or that require different database connections.

---

### Creating a ChUI project

To create an OpenEdge project specialized for character-based UI development:

1. Select **File > New > OpenEdge Project** from the Progress Developer Studio for OpenEdge main menu bar or the Project Explorer view context (right-click) menu. The New OpenEdge Project wizard opens and displays the *Create an OpenEdge Project page* on page 105.

2. Type a name in the **Project name** field.

   The root folder for the new project is created in your current Eclipse workspace, unless you specify a different location.

3. To choose a different location for the project, click and clear the **Use default** check box. Then either browse to, or enter the path to the desired location.

4. Select **Client** from the left section.

5. Select the **Web Application** option button and select the **ChUI** option button.

6. Click **Next**. The *Select AVM and layout options page* on page 107 appears.

7. Specify the OpenEdge AVM options and the Project layout options (or leave the default settings) and click **Next**. The *Define PROPATH page* on page 109 appears.

8. Leave the default settings or modify the PROPATH entries to be used by the project and click **Next**. The *Select database connections page* on page 109 appears.

9. Select the database connections to be used by the project and click **Finish**.

You will see your project and its resources in the **Project Explorer** view.

**See also**

*Project types in Progress Developer Studio for OpenEdge* on page 86

*Creating a new OpenEdge project*
Creating a GUI for .NET project (Windows only)

To create an OpenEdge project specialized for .NET-based GUI development:

1. Select **File > New > OpenEdge Project** from the Progress Developer Studio for OpenEdge main menu bar or the Project Explorer view context (right-click) menu. The New OpenEdge Project wizard opens and displays the *Create an OpenEdge Project* page on page 105.

2. Type a name in the **Project name** field.

   The root folder for the new project is created in your current Eclipse workspace, unless you specify a different location.

3. To choose a different location for the project, click and clear the **Use default** check box. Then either browse to, or enter the path to the desired location.

4. Select **Client** from the left section.

5. Select the **Desktop Application** option button and select the **GUI for .NET** option button.

6. Click **Next**. The *Select AVM and layout options* page on page 107 appears.

7. Specify the OpenEdge AVM options and the Project layout options (or leave the default settings) and click **Next**. The *Define PROPATH* page on page 109 appears.

8. Leave the default settings or modify the PROPATH entries to be used by the project and click **Next**. The *Select database connections* page on page 109 appears.

9. Select the database connections to be used by the project and click **Finish**.

You will see your project and its resources in the **Project Explorer** view.

See also

- Project types in Progress Developer Studio for OpenEdge on page 86
- Creating a new OpenEdge project
- Create an OpenEdge Project page on page 105
- Project Facets page on page 111
- Select AVM and layout options page on page 107
- Define PROPATH page on page 109
- Select database connections page on page 109

Creating a standard OpenEdge project

To create a standard OpenEdge project without specialized properties:

1. Select **File > New > OpenEdge Project** from the Progress Developer Studio for OpenEdge main menu bar or the Project Explorer view context (right-click) menu. The New OpenEdge Project wizard opens and displays the *Create an OpenEdge Project* page on page 105.

2. Type a name in the **Project name** field.

   The root folder for the new project is created in your current Eclipse workspace, unless you specify a different location.
3. To choose a different location for the project, click and clear the **Use default** check box. Then either browse to, or enter the path to the desired location.

4. Select **General** from the left section.

   **Note:** If you do not select a category (Client, Server, or General) and click **Finish**, a basic OpenEdge project without any specialization is created.

5. Select the **OpenEdge Basic** option button.

6. Click **Next**. The **Select AVM and layout options page** on page 107 appears.

7. Specify the OpenEdge AVM options and the Project layout options (or leave the default settings) and click **Next**. The **Define PROPATH page** on page 109 appears.

8. Leave the default settings or modify the PROPATH entries to be used by the project and click **Next**. The **Select database connections page** on page 109 appears.

9. Select the database connections to be used by the project and click **Finish**.

You will see your project and its resources in the **Project Explorer** view.

**See also**

- Project types in Progress Developer Studio for OpenEdge on page 86
- Creating a new OpenEdge project
- Create an OpenEdge Project page on page 105
- Project Facets page on page 111
- Select AVM and layout options page on page 107
- Define PROPATH page on page 109
- Select database connections page on page 109

**Creating a custom project**

You can create a new custom project by following the below steps or you can select one of the predefined custom project configurations (if any) that appear in the **Custom** option in the **General** category in the **OpenEdge Project** wizard.

When you select a predefined custom project, the **New OpenEdge** wizard displays the folder structure of the project on the **Review custom project layout page** on page 111 and the PROPATH entries are displayed on the **Define PROPATH page** on page 109.

To create a custom project:

1. Select **File > New > OpenEdge Project** from the Progress Developer Studio for OpenEdge main menu bar or the Project Explorer view context (right-click) menu. The New OpenEdge Project wizard opens and displays the **Create an OpenEdge Project page** on page 105.

2. Type a name in the **Project name** field.
   
   The root folder for the new project is created in your current Eclipse workspace, unless you specify a different location.

3. To choose a different location for the project, click and clear the **Use default** check box. Then either browse to, or enter the path to the desired location.

4. Select **General** from the left section.

5. Select the **Custom** option button.
6. Click **Next**. The **Project Facets page** on page 111 appears.

7. Select the facets to be enabled for the project or leave the default settings and click **Next**. The **Select AVM and layout options page** on page 107 appears.

8. Specify the OpenEdge AVM options and the Project layout options (or leave the default settings) and click **Next**. The **Define PROPATH page** on page 109 appears.

9. Leave the default settings or modify the PROPATH entries to be used by the project and click **Next**. The **Select database connections page** on page 109 appears.

10. Select the database connections to be used by the project and click **Finish**.

You will see your project and its resources in the **Project Explorer** view.

**See also**
Project types in Progress Developer Studio for OpenEdge on page 86
Creating a new OpenEdge project
Create an OpenEdge Project page on page 105
Project Facets page on page 111
Select AVM and layout options page on page 107
Define PROPATH page on page 109
Select database connections page on page 109

**Creating a Dynamics project (Windows only)**

To create an OpenEdge project specialized for development of applications incorporating Progress Dynamics:

1. Select **File > New > OpenEdge Project** from the Progress Developer Studio for OpenEdge main menu bar or the Project Explorer view context (right-click) menu. The New OpenEdge Project wizard opens and displays the **Create an OpenEdge Project page** on page 105.

2. Type a name in the **Project name** field.
   
   The root folder for the new project is created in your current Eclipse workspace, unless you specify a different location.

3. To choose a different location for the project, click and clear the **Use default** check box. Then either browse to, or enter the path to the desired location.

4. Select **General** from the left section.

5. Select the **Dynamics** option button.

6. Click **Next**. The **Select AVM and layout options page** on page 107 appears.

7. Specify the OpenEdge AVM options and the Project layout options (or leave the default settings) and click **Next**. The **Define Dynamics configuration page** (Windows only) on page 110 appears.

8. Specify the configuration file and session type to be used by the project and click **Next**. The **Define PROPATH page** on page 109 appears.

9. Leave the default settings or modify the PROPATH entries to be used by the project and click **Next**. The **Select database connections page** on page 109 appears.

10. Select the database connections to be used by the project and click **Finish**.

You will see your project and its resources in the **Project Explorer** view.

**See also**
Project types in Progress Developer Studio for OpenEdge on page 86
Creating an AppBuilder project

To create an OpenEdge project specialized for AppBuilder application development:

1. Select **File > New > OpenEdge Project** from the Progress Developer Studio for OpenEdge main menu bar or the Project Explorer view context (right-click) menu. The New OpenEdge Project wizard opens and displays the Create an OpenEdge Project page on page 105.

2. Type a name in the **Project name** field.
   The root folder for the new project is created in your current Eclipse workspace, unless you specify a different location.

3. To choose a different location for the project, click and clear the **Use default** check box. Then either browse to, or enter the path to the desired location.

4. Select **Client** from the left section.

5. Select the **Desktop Application** option button and select the **AppBuilder** option button.

6. Click **Next**. The Select AVM and layout options page on page 107 appears.

7. Specify the OpenEdge AVM options and the Project layout options (or leave the default settings) and click **Next**. The Define PROPATH page on page 109 appears.

8. Leave the default settings or modify the PROPATH entries to be used by the project and click **Next**. The Select database connections page on page 109 appears.

9. Select the database connections to be used by the project and click **Finish**.

You will see your project and its resources in the **Project Explorer** view.

See also

Project types in Progress Developer Studio for OpenEdge on page 86
Creating a new OpenEdge project
Create an OpenEdge Project page on page 105
Project Facets page on page 111
Select AVM and layout options page on page 107
Define PROPATH page on page 109
Select database connections page on page 109

Adding resources to a project

Resources are the files and folders that comprise a project.

**Note:** For information about adding existing resources to a project see Exporting and importing resources and settings.

To add a new file or folder to an OpenEdge project:
1. Select the project name from the **Project Explorer view**.

2. Right-click and select **New** from the pop-up menu.

3. Select either Folder or one of the ABL file types from the **New** menu:

   ![ABL Class
   ABL Interface
   ABL Procedure
   ABL Include
   Folder
   File
   ABL Form
   ABL Dialog
   ABL MDI Form
   ABL User Control
   ABL Inherited Control]

4. If you select **Folder**, the **New Folder** dialog appears. Specify the location of the folder in the project tree and name it.

   If you want to add a folder that already exists in your file system, click the **Advanced** button. Select **Link** to folder in the file system and browse to find the folder.

5. If you select one of the ABL file types, a dialog appears that allows you to specify the name and other information about the file.

   The resulting file is generated from a standard ABL template. Therefore, the file will contain sections appropriate to the file type, and the file will have an appropriate extension. For example, **ABL Procedure** produces a .p file similar to the following:

   ```
   /***************************************************************************/
   File : query.p
   Purpose :
   Syntax :
   Description :
   Author(s) :
   Created : Tue Aug 22 11:29:25 EDT 2006
   Notes :
   /***************************************************************************/
   /***************************************************************************/
   /***************************************************************************/
   /***************************************************************************/
   /***************************************************************************/
   ```

6. If you select **File**, you can link to a file in the local file system.

   In the **New File** dialog, click the **Advanced** button. Select **Link** to folder in the file system and browse to find the file that you want to include in your project.
Adding existing database connections to a project

You can either add an existing database connection to your project, or you can create a new database connection for your project. An existing database connection is one that can be shared by all the projects in your workspace. It appears in the Database Connections project property page.

To add to an existing database connection:

1. From the Project Explorer view, select a project.
2. Right-click to expand the context menu.
3. From the context menu, select Properties.
4. In the Project Properties page, expand Progress OpenEdge in the tree view on the left.
5. Select Database Connections.
6. Select the Show All radio button to list all connected databases.
7. Click check boxes next to the databases that you want to add to the project.
8. Click OK.

Adding a new database connection to a project

To create a new database connection:

1. From the Project Explorer view, select a project.
2. Right-click to expand the context menu.
3. From the context menu, select Properties.
4. In the Project Properties page, expand Progress OpenEdge in the tree view on the left.
5. Select Database Connections.
6. Select Configure Database Connections from the top of the Database Connections properties page.
7. Click New to open the Add Connection Profile wizard.

8. Specify connection parameters on pages of the Add Connection Profile wizard. When you complete the wizard, the new connection is listed in the Database Connections preferences page. Click the checkbox next to the connection name to add it to the project.

See also
Creating a new OpenEdge project
Adding new resources to a project on page 95
Adding existing database connections to a project on page 97
Setting project properties on page 98
Exporting and importing project resources and properties on page 100
Database Connections properties page on page 120

Setting project properties

To review the project properties and modify them if necessary:

1. Right-click the project in the Project Explorer view and select Properties from the context menu.
2. In the left pane of the Properties page, expand Progress OpenEdge in the tree view.
3. Click the appropriate node and make any desired changes. Click Apply to save the settings on the currently open tab and continue, or click OK to save all settings and close the Project Properties dialog.

Note: If you previously saved project settings as a named profile, and those settings are appropriate for the current project, you can load them by clicking Change Profile on the Progress OpenEdge page and selecting the profile.

For detailed information about the options on all property pages, see Project property settings.

See also
Creating a new OpenEdge project
Adding resources to a project on page 95
Adding existing database connections to a project on page 97
Adding a new database connection to a project on page 97
Exporting and importing project resources and properties on page 100
Configuration variables on page 84
Saving project property profiles

You can save your current project property settings as a named project property profile. This allows you easily change from one group of settings to another. Project property profiles are available to any new or existing project in your workspace.

**Note:** Changing project property profiles does not change PROPATH settings, source and build folders, or database connections for a project. These settings are not saved in a project property profile.

To save a project property profile:

1. Right-click the project folder in the **Project Explorer** view and select **Properties** from the context menu.
2. In the left pane of the **Properties** window, select **Progress OpenEdge**.
3. Select the **Custom** page and click **Save Profile**.
4. Enter a name for the profile and click **OK**.

**See also**
- Setting project properties on page 98
- Sharing project settings on page 100
- Changing project property profiles on page 99
- Progress OpenEdge project properties_pages on page 111
- New OpenEdge Project wizard on page 105

Changing project property profiles

To change to a different project property profile:

1. Right-click the project folder in the **Project Explorer** view and select **Properties** from the context menu.
2. In the left pane of the **Properties** window, select **Progress OpenEdge**.
3. Click **Change Profile**.
4. Select a profile from the **Select Development Profile** dialog.
5. Click **OK**. A message confirms that the selected profile was added to the project.

**See also**
- Setting project properties on page 98
- Sharing project settings on page 100
- Saving project property profiles on page 99
- Progress OpenEdge project properties_pages on page 111
**Exporting and importing project resources and properties**

A generic feature of Eclipse is the ability to import resources and settings from one project to another. The topics in this section describe the options for importing resources and settings.

**Sharing project properties**

You can store the current set of project properties so that they are included with the rest of the content when the project is imported into a different workspace. This eliminates the need to reset the properties.

To share the current project settings:

1. Right-click the project folder in the **Project Explorer** view and select **Properties** from the context menu.
2. In the left pane of the **Properties** window, select **Progress OpenEdge > Custom**.
3. On the **Custom** tab, click **Share Settings**.

Sharing settings saves the property settings in a file called `project.xml`, located in the project's root directory.

---

**Note:** The `project.xml` file does not contain PROPATH settings, source and build folders, or database connection information. PROPATH settings and source/build folder settings are saved in the project's `.propath` file. Database connection settings are saved in the project's `.dbconnection` file.

If `project.xml` file is not present when a project is imported its properties revert to default settings.

**See also**

- Setting project properties on page 98
- Saving a named project profile on page 99
- Saving a named project profile on page 99
- Custom settings on page 119

**Importing resources**

There are several methods to import resources (files and folders) into a project:

1. Use the context menu in the **Project Explorer** view. The context menu contains the typical file system commands (such as Copy and Paste). By navigating in the Project Explorer tree view, you can use these commands to copy or move file resources from one project to another.

2. Use the import (**File > Import > General > File System**) or export (**File > Export > General > File System**) wizard to manage file and folder resources.

3. In the **New File** or **New Folder** dialog (see Adding new resources to a project), click the **Advanced** button. Select Link to folder (or File) in the file system and browse to find the file that you want to include in your project.

---

**Note:** When you copy a file from one project to another, any breakpoints that existed in the original are not preserved in the copy. However, you can export breakpoints to an XML file and then import breakpoints to a file in another project.
Exporting projects

You can use basic Eclipse Export/Import functionality to create an archive of a project that can be distributed to other systems or to other users.

To export a project:

1. Do the following before exporting a project:
   a) Click **Share Settings** on the Custom properties sheet to create a project.xml file in your project folder.
   b) Export any workspace level database connection profiles using the **Export Workspace Database Connections** dialog. The profile information goes to an XML file which you must import separately from the project archive.

2. To create an archive of the project:
   a) Select **File > Export > General > Archival File**.
   b) Select the project name and all project files in the **Archive File** dialog. Be sure to include all "dot" files (.propath, .dbconnection, etc.) and the **project.xml** file.
   c) Specify the path and the name of the archive file.
   d) Select the archive format (tar or zip) and any other options you require.
   e) Click **Finish**.

Importing projects

You can use basic Eclipse Export/Import functionality to create an archive of a project that can be distributed to other systems or to other users.

To import an archived project:

1. Start Progress Developer Studio for OpenEdge in the workspace where you want to import the project.
2. **Import database connection profiles**.
3. Select **File > Import > General > Existing Projects into Workspace**.
4. On the **Import Projects** page, choose **Select archive file**.
5. Browse to the archive file that contains the project.
6. Click **Finish**. The project is created and opened in the workspace. Note, however, that **breakpoints** are not preserved.
See also
Exporting projects on page 101
Sharing project properties on page 100
Exporting database connections on page 102
importing database connections on page 102
Importing breakpoints on page 103

Exporting database connections
To export connection information:

1. Start Progress Developer Studio for OpenEdge in the workspace that is the source of the connection information.

2. Select File > Export. The Export dialog appears.

3. Choose the Progress OpenEdge/Database Connections node and click Next. The Export Workspace Database Connections dialog appears.

4. Select the database connections you want to export by clicking the checkbox next to the connection name. You can export all connections by selecting the Select All radio button.

5. In the To database connection file field, specify the target location for the exported XML file.

6. Select the Overwrite existing files without warning checkbox to replace existing database files without a warning message.

7. Select the Exclude user ID and password checkbox to prevent user ID or password information from being included in the exported XML file. You should use this option if you are exporting connection information to some other user.

8. Click Finish.

See also
Database Connections preferences on page 120
Adding existing database connections to a project on page 127
Adding a new database connection to a project on page 124
Importing database connections on page 102

Importing database connections
You can import database connection information from one workspace to another. This eliminates the task of manually duplicating connection profiles. First you export connection information from a workspace to an XML file. Then, you can import that XML file to another workspace.

To import connection information:

1. Start Progress Developer Studio for OpenEdge in the workspace where the connection information will be imported.

2. Select File > Import. The Import dialog appears.

3. Choose the Progress OpenEdge/Database Connections node. The Import Workspace Database Connections dialog appears.
4. In the **From database connection file** field, enter the name of the database connection file that you want to import. The database connection file is an XML file created in the **Export Workspace Database Connections** dialog.

5. Select the **Ignore duplicate connections** checkbox to suppress the listing of connections that already exist in your workspace. The Database connections browse shows a list of the connections that are defined in the database connection file. You can select any or all of the connections from the list.

6. Click **Finish** and the imported connections will be available to the projects in your workspace.

**See also**
- Database Connections preferences on page 120
- Adding existing database connections to a project on page 127
- Adding a new database connection to a project on page 124
- Exporting database connections on page 102

**Importing breakpoints**

When you copy a project from one workspace to another, any breakpoints that existed in the original are not preserved in the copy. However, you can export breakpoints from a workspace to an XML file and then import those breakpoints to projects in another workspace.

To import breakpoints from one workspace to another:

1. Start Progress Developer Studio for OpenEdge in the workspace that is the source of the breakpoint information.
2. Choose **File > Export > General > Breakpoints**.
3. Click **Next** and the **Export Breakpoints** dialog appears.
4. Select the breakpoints you want to export from the checklist. Breakpoints from files in all open projects are listed.
5. Specify the path name of the file where you want to export the breakpoints.
6. Click **Finish** and the breakpoints are saved to an XML file. By default, a `.bkpt` extension is added to the file name.
7. Start Progress Developer Studio for OpenEdge in the workspace where you want to import the breakpoint information.
8. Choose **File > Import > General > Breakpoints**.
9. Click **Next** and the **Import Breakpoints** dialog appears.
10. Browse and select the breakpoint file specified in Step 4 on page 103.
11. Click **Finish**. The breakpoints are added to the appropriate files.

**Note:** Breakpoints are only imported for existing resources. The project and the actual resource file must exist in the current workspace before you can import breakpoints. No warning message is displayed when the import fails because the resource does not exist.
Migrating database connections

Database connections created in Release 10.1A cannot be used in later releases of Progress Developer Studio for OpenEdge. If you want to use a database connection created in Release 10.1A in this release of Progress Developer Studio for OpenEdge, it must be migrated using the database migration function. The other alternative is to re-create the database connection using the tools in the current release.

When you migrate database connections, Progress Developer Studio for OpenEdge extracts the old database connection property settings and creates a new database resource file using the existing connection information. The revised database connections are automatically mapped to the current project in the workspace. Old database property settings are removed for a project once it is migrated. Each project can only be migrated once.

To migrate database connections:

1. Select the OpenEdge Editor perspective.
2. From the menu bar, select OpenEdge.
3. Select Migration > Migrate 10.1A Database Connections. The Migrate Database Connections dialog appears.
4. Select the project (or projects) that have database connections you want to migrate.
5. Click Next to display existing projects and their associated database connections.

Note: If multiple connections have the same name, only the first instance is selected for migration. That migrated connection will be automatically assigned to every project in which the name occurs. If you select additional instances for migration, an identifier (for example, -1) will be appended to form a unique name for each instance. Select multiple instances when the connection definitions differ.

6. Click Finish.

Defining new configuration variables

In addition to the predefined variables available in Progress Developer Studio for OpenEdge, you can define new configuration variables at the project and workspace level. Creating configuration variables at the project level ensures that the projects, which contain linked resources using the configuration variables, remain portable across workspace and systems.

To define new configuration variables:

1. Right-click the project in the Project Explorer view and select Properties from the context menu.
2. Expand Progress OpenEdge in the tree view in the left pane of the Properties page.
3. Select the Configuration Variables node.
4. Click Add to open the New Variable dialog.
5. Specify a name for the new variable in the Name field.
6. Specify an associated location for the new variable. You can provide a directory location on the file system by clicking Folder or select an existing variable by clicking Variable.
7. Click OK to save the settings.

To edit an existing configuration variable, select the variable and click Edit on the Configuration Variables properties page.
New OpenEdge Project wizard

The New OpenEdge Project wizard allows you to create faceted OpenEdge projects in a simple and configurable manner. It contains multiple pages: the pages vary depending on the type of project you select on the Create an OpenEdge Project page on page 105.

You can start the New OpenEdge Project wizard by selecting File > New > OpenEdge Project from the main menu bar.

Based on the project type you select on the Create an OpenEdge Project page on page 105, the New OpenEdge Project wizard displays the appropriate page. This section documents these pages.

See also
Creating a new OpenEdge Project

Create an OpenEdge Project page

The Create an OpenEdge Project page of the New OpenEdge Project wizard allows you to select the type of faceted project you want to create and set general options for it.

Note: The subsequent pages on the New OpenEdge Project wizard depend on the project type you select on this page.

It contains the following controls:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project name</td>
<td>Allows you to assign a name that is unique in the workspace to a folder that will contain project resources.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Avoid changing the project name outside of Progress Developer Studio for OpenEdge (with Windows Explorer, for example). The project will become corrupted and you will be unable to reopen it.</td>
</tr>
<tr>
<td></td>
<td>To change the name after the project is created, choose the project name in the Project Explorer view and select <strong>File &gt; Rename</strong> from the main menu bar.</td>
</tr>
<tr>
<td>Use default location</td>
<td>Specifies that the project folder will be created in the default location, which is the top level of the workspace folder.</td>
</tr>
<tr>
<td>Location</td>
<td>When Use default location is not selected, specifies the directory that will contain the project folder.</td>
</tr>
<tr>
<td>Project type configuration</td>
<td>Allows you to select the project type you want to create. The default value is OpenEdge.</td>
</tr>
<tr>
<td>Next</td>
<td>Shows the next page of the wizard based on the project type you select.</td>
</tr>
<tr>
<td>Finish</td>
<td>Creates a project with the default configuration settings defined for the selected project type.</td>
</tr>
</tbody>
</table>
The **Project type configuration** drop-down menu provides the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;custom&gt;</td>
<td>Creates a custom project that is a combination of different project types added to a single project.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>When you create a custom project type you must select the facets associated with the project on the <strong>Project Facets page</strong> on page 111.</td>
</tr>
<tr>
<td>REST</td>
<td>Creates an OpenEdge project with REST service functionality.</td>
</tr>
<tr>
<td>AppServer</td>
<td>Creates an OpenEdge project with AppServer functionality.</td>
</tr>
<tr>
<td>ChUI</td>
<td>Creates an OpenEdge project with character-based UI functionality.</td>
</tr>
<tr>
<td>Dynamics</td>
<td>Creates an OpenEdge project for Progress Dynamics application development.</td>
</tr>
<tr>
<td>GUI for .NET</td>
<td>Creates an OpenEdge project with .NET-based GUI.</td>
</tr>
<tr>
<td>OpenEdge</td>
<td>Creates a standard OpenEdge project.</td>
</tr>
<tr>
<td>WebSpeed</td>
<td>Creates an OpenEdge project with WebSpeed functionality for traditional servers.</td>
</tr>
<tr>
<td>Data Object</td>
<td>Creates an OpenEdge project for Data Object application development to be deployed to traditional servers.</td>
</tr>
<tr>
<td>ABL Web App</td>
<td>Creates an OpenEdge project for deploying one or more ABL services or static web user interface content as a single web app to Progress Application Server for OpenEdge.</td>
</tr>
</tbody>
</table>

**Note:** The **Project type configuration** drop-down menu also displays the custom projects that you create using the Customization Editor. For more information see Customization help.

### See also
- Project types in Progress Developer Studio for OpenEdge on page 86
- OpenEdge project and resource data files on page 83
- Creating a new OpenEdge project
- Select AVM and layout options page on page 107
- Define AppServer content module page on page 108
- Define PROPATH page on page 109
- Select database connections page on page 109
- Define WebSpeed dynamic content module page on page 109
- Define Dynamics configuration page Windows only on page 110
- Review custom project layout page on page 111
- Project Facets page on page 111
- Progress OpenEdge project properties pages on page 111
Select AVM and layout options page

The **Select AVM and layout options** page of the **New OpenEdge Project** wizard allows you to specify the OpenEdge AVM options and the Project layout options. You can use this page to specify the project runtime, the project source folder, and the project r-code folder. The OpenEdge AVM options on this page depend upon the project type you select on the **Create an OpenEdge Project page** on page 105 and the facets associated with it.

It contains the following controls:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use project-specific AVM</strong></td>
<td>Specifies that an AVM, dedicated solely to the project you are creating, starts whenever you open the current workspace. To change this setting after the project is created, use the Progress OpenEdge properties page.</td>
</tr>
<tr>
<td><strong>Use shared AVM</strong></td>
<td>Specifies use of the AVM that is available to all projects in the workspace. To change this setting after the project is created, use the Progress OpenEdge properties page.</td>
</tr>
<tr>
<td><strong>Use TTY for runtime</strong></td>
<td>Allows you to run ABL procedures and classes in a character client window rather than in a GUI window. A separate window opens for each program that you run. The wizard automatically selects this option based on the project type you select on the <strong>Create an OpenEdge Project page</strong> on page 105.</td>
</tr>
</tbody>
</table>

**Note:** The GUI client is supported only on Windows. Therefore, on non-Windows platforms, **Use TTY for runtime** is automatically selected and cannot be changed. It does not change according to the type of project you select.

In addition to the client windows that open when you run programs, a runtime console is launched for the project AVM when the project opens or for the shared AVM when the workspace opens. However, by default this console window is hidden.

After the project is created, to change the TTY setting or to make the project runtime console visible, use the Progress OpenEdge properties page (if using a project-specific AVM) or the Shared AVM preferences page.
Specifiesthat source and r-code will be in the top-level of the project folder.

**Note:** This option is selected by default.

Specifiesthat source and r-code are located if the location is other than the project root directory. Paths must be specified relative to the project root directory.

**Note:** This setting establishes a relationship only between one source location and specified corresponding r-code destination. If r-code is generated from source residing in any other location in the project, it will end up, by default, in the same directory as the source. Also note that the default build destination property can be changed if you clear the Source directory selection but retain the R-code directory selection. The specified r-code destination will be applicable to source code from any location in the project.

After the project is created, use the Build or PROPATH/Source properties pages to modify or extend these settings.

---

**See also**
- OpenEdge project and resource data files on page 83
- Creating a new OpenEdge project
- Create an OpenEdge Project page on page 105
- Define AppServer content module page on page 108
- Define PROPATH page on page 109
- Select database connections page on page 109
- Define WebSpeed dynamic content module page on page 109
- Define Dynamics configuration page Windows only on page 110
- Review custom project layout page on page 111
- Project Facets page on page 111
- Progress OpenEdge project properties pages on page 111

**Define AppServer content module page**

The Define AppServer content module page of the New OpenEdge Project wizard appears only if you select an AppServer project type on the Create an OpenEdge Project page on page 105. It allows you to define the project module and choose publishing options.

The following controls are available on this page:

<table>
<thead>
<tr>
<th>Module name</th>
<th>Allows you to specify the name of the module. The default is to use the same name as the project.</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppServer source folder</td>
<td>Allows you to specify or create the folder that contains the files that will be published to the server. The default is to create a folder named AppServerContent in the project folder.</td>
</tr>
<tr>
<td>Supported servers</td>
<td>Allows you to select where to publish the files, from a list of available servers.</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Publish changes immediately</td>
<td>Allows you to choose if you want to copy files to the server immediately after the project is created.</td>
</tr>
</tbody>
</table>

**See also**

Creating a new OpenEdge project

**Define PROPATH page**

The Define PROPATH page of the New OpenEdge Project wizard allows you to view and modify the PROPATH entries configured in the project type that you select on the Create an OpenEdge Project page on page 105.

By default all PROPATH entries are part of the OpenEdge tooling. To exclude a PROPATH entry from OpenEdge tooling, select the **Exclude from OpenEdge tooling** node under the PROPATH entry that you want to exclude. The **Exclude from OpenEdge tooling** node appears as a child node for each PROPATH entry.

If the project types you select provide PROPATH entries, then the order of the PROPATH entries depends on the hierarchy of those project types.

After the project is created, you can change PROPATH settings from the PROPATH property page. However, if the project uses a shared AVM, you must use the Shared AVM/PROPATH preference page.

**See also**

PROPATH Properties page on page 122

**Select database connections page**

The Select database connections page of the New OpenEdge Project wizard allows you to choose from the available database connection profiles in the workspace. It also allows you to start the Progress Developer Studio for OpenEdge database connection wizard, which you can use to create a new database connection profile.

After the project is created, you can add or remove database connection profiles from the Database connections properties page. However, if the project uses the shared AVM, you must use the Shared AVM/Database Connections preference page.

**See also**

Adding a new database connection to a project on page 97

Database Connections properties page on page 120

**Define WebSpeed dynamic content module page**

The Define WebSpeed dynamic content module page of the New OpenEdge Project wizard appears only if you select a WebSpeed project type on the Create an OpenEdge Project page on page 105. It allows you to define the WebSpeed project module and choose publishing options.

The following controls are available on this page:
### Define Dynamics configuration page (Windows only)

The **Define Dynamics configuration** page of the **New OpenEdge Project** wizard allows you to select a configuration file for Progress Dynamics application development. It appears when you select the Dynamics project type on the **Create an OpenEdge Project page** on page 105.

The following controls are available on this page:

<table>
<thead>
<tr>
<th>Configuration file</th>
<th>Allows you to specify the path of the XML configuration file where Dynamics session types are defined. The default path is %DLC%/gui/dynamics/icfconfig.xml.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session type</td>
<td>Allows you to specify the name of the session type. The default value is: <strong>ICFDev</strong>.</td>
</tr>
</tbody>
</table>

**Note:** You can view the PROPATH entries for the Dynamics project type folder layout on the **Define PROPATH page** of the **New OpenEdge Project** wizard.

### See also

- Project types in Progress Developer Studio for OpenEdge on page 86
- Creating a Dynamics project (Windows only) on page 94
- New OpenEdge Project wizard on page 105

---

<table>
<thead>
<tr>
<th>Module name</th>
<th>Allows you to specify the name of the module. The default is to use the same name as the project.</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebSpeed source folder</td>
<td>Allows you to specify or create a folder that contains the files to be published to the server. By default, a folder named <strong>DynamicContent</strong> is created in the project folder.</td>
</tr>
<tr>
<td>Supported servers</td>
<td>Allows you to select where to publish the files, from a list of available servers.</td>
</tr>
<tr>
<td>Publish changes immediately</td>
<td>Allows you to specify if you want to publish files to the server immediately after the project is created.</td>
</tr>
<tr>
<td>SpeedScript .w file destination folder</td>
<td>Allows you to specify or create the folder that will contain the .w file that is generated by the SpeedScript Editor. By default, a folder named <strong>SpeedScriptGen</strong> is created. You can use this to debug SpeedScript files. For more information on debugging SpeedScript files, see Running and Debugging ABL Programs</td>
</tr>
</tbody>
</table>
Review custom project layout page

The Review custom project layout page of the New OpenEdge Project wizard displays the folder structure of the custom project type which you select on the Create an OpenEdge Project page on page 105. The content shown on this page is read-only. The PROPATH entries of the custom project type folder layout are displayed in the Define PROPATH page on page 109.

Note: The Create an OpenEdge Project page displays all the custom project types that you create using the Customization Editor. See Customization help for information on creating custom project types.

When you select multiple custom project types that have a folder structure with similar files, the page prompts you if you want to override the files and display a single instance of the files.

See also
Project types in Progress Developer Studio for OpenEdge on page 86
Creating a custom project on page 93
New OpenEdge Project wizard on page 105

Project Facets page

The Project Facets page of the New OpenEdge Project wizard allows you to view, add, and remove facets associated with the custom project type configuration that you select on the Create an OpenEdge Project page on page 105.

Note: Progress Developer Studio for OpenEdge allows you to create a custom project type when you want a combination of different project types to be added to a single project. To do so, select <custom> from the Project Type drop-down list on the Create an OpenEdge Project page on page 105.

See also
Project types in Progress Developer Studio for OpenEdge on page 86
Creating a custom project on page 93
New OpenEdge Project wizard on page 105

Progress OpenEdge project properties pages

The Properties dialog is where you specify the values that control the behavior of the currently selected project. This dialog appears when you select a project folder in the Project Explorer view and select Properties from the context menu, the Project menu (Project > Properties), or the File menu (File > Properties).

In a tree view on the left of the window, you can select project properties pages that apply to the Eclipse framework, to third party plugins, or to Progress Developer Studio for OpenEdge. The Progress OpenEdge node contains project properties pages that apply specifically to OpenEdge projects.

See also
Project types in Progress Developer Studio for OpenEdge on page 84
Creating a new OpenEdge project
Setting project properties on page 98
Progress OpenEdge properties page

The Progress OpenEdge properties page allows you to view and to adjust certain characteristics of the OpenEdge AVM, which is the client that runs your ABL code.

To access the Progress OpenEdge properties page, right-click an OpenEdge project name in the Project Explorer view and choose Properties from the context menu. Click the Progress OpenEdge node to view the page.

The following controls are available:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change Profile</td>
<td>Load a previously saved set of project properties (see Custom settings). The current profile appears to the right of this button. Click to select from a list of saved profiles.</td>
</tr>
<tr>
<td>Use OpenEdge shared AVM</td>
<td>Use the AVM that is available to all projects in the workspace rather than an AVM that is dedicated to the currently selected project. If you change an existing project to use the shared AVM, you may need to update the shared AVM settings to include the PROPATH, assemblies, and database connections of the project.</td>
</tr>
<tr>
<td>Note:</td>
<td>If you select the shared AVM for a project, most of the settings on this property page will appear dimmed. They cannot be changed on this page, and they may also be inaccurate. You must go to the Shared OpenEdge AVM preference page to view or to change those settings. Select Window &gt; Preferences from the main menu bar. Then open Progress OpenEdge &gt; Shared OpenEdge AVM from the Preferences page tree view. Also be aware that settings changed on the Shared OpenEdge AVM preference page affect all projects that use the shared AVM.</td>
</tr>
<tr>
<td>Configure shared OpenEdge AVM</td>
<td>Open the Shared OpenEdge AVM preference page, which allows you to change shared AVM settings.</td>
</tr>
<tr>
<td>Current version</td>
<td>The version of the OpenEdge runtime that is used by Progress Developer Studio for OpenEdge (a read-only informational field).</td>
</tr>
<tr>
<td>Installation</td>
<td>The root directory in which OpenEdge is installed (a read-only informational field).</td>
</tr>
<tr>
<td>Working directory</td>
<td>Specify the directory in which the OpenEdge runtime starts when the project is opened. By default, this is the project root directory.</td>
</tr>
<tr>
<td>Temp directory</td>
<td>Specify the directory in which temporary files are created. By default, this is the working directory specified when Progress Developer Studio for OpenEdge was installed.</td>
</tr>
</tbody>
</table>
Specify any appropriate parameters for starting the AVM in the context of the currently selected project.

Refer to the ABL parameters section of the OpenEdge Deployment: Startup Command and Parameter Reference for a list of AVM (also called the ABL client) startup parameters. You can find the manual in the Product Documentation section of the Progress Software Developer's Network Web site.

**Note:** Do not specify database connections as startup parameters. Use the Database Connections page for this purpose.

You can use a parameter file by including `-pf parameter_file_path`. A relative path statement is interpreted relative to the project working directory.

If you check the **Add default parameters** option, the parameters in the **Startup parameters** box are appended to the defaults.

| **Startup parameters** | **Specify any appropriate parameters for starting the AVM in the context of the currently selected project.**
| | **Note:** Do not specify database connections as startup parameters. Use the Database Connections page for this purpose.
| | **You can use a parameter file by including `-pf parameter_file_path`. A relative path statement is interpreted relative to the project working directory.**
| | **If you check the **Add default parameters** option, the parameters in the **Startup parameters** box are appended to the defaults.**
| **Add default parameters** | **An option to use the default startup parameters.**
| | **Note:** Although you can see the default startup parameters on this page, you cannot change them here. Go to the Progress OpenEdge/Startup preference page to change the defaults.
| **Use TTY for runtime** | **An option to run ABL procedures and classes in a character client window rather than in a GUI window. A separate window opens for each program that you run.**
| | **Note:** The GUI option is supported only on Windows. Therefore, on non-Windows platforms, **Use TTY for runtime** is automatically selected and unchangeable.
| | **In addition to the windows that open when you run a program, a runtime console is launched for the project-specific AVM when you open the project. However, by default this console window is hidden.**
| **Hide TTY runtime console** | **An option, selected by default, to suppress the display of the project-specific runtime console. This setting is changeable only if **Use shared AVM** is deselected and **Use TTY for Runtime** is selected.**
| **Use global Toolbox for Visual Designer** | **An option to use a global (common) Toolbox that might be shared with other projects in the workspace. By default, each OpenEdge project that you create uses a dedicated Toolbox that is specific to that project.**
| **Share Settings** | **Saves the current set of project properties in a project.xml file in the project root directory, allowing the project to retain its properties after being imported to another workspace.**
| **Import Settings** | **Imports a set of project properties from a previously generated properties file and applies them to the currently selected project.**
| **Save Profile** | **Saves the current set of project properties so that you can recall them and apply them to any project. You can load a saved profile on the Progress OpenEdge properties page.**
The AppBuilder properties page allows you to configure the properties of your AppBuilder or Progress Dynamics project in Progress Developer Studio for OpenEdge. This controls the behavior of the ABL GUI procedure (.w) files in the currently selected OpenEdge or Dynamics project.

You can access the AppBuilder page by right-clicking an OpenEdge project name in the Project Explorer view and choosing Properties from the context menu. Expand the Progress OpenEdge node in the tree view, and then click the AppBuilder node.

**Note:** If you chose the shared AVM for the currently selected project, most of the settings on this property page will be disabled. They cannot be changed on this page. You must go to the AppBuilder page under the Shared OpenEdge AVM preference page to view or to change these settings. See *Setting up a Shared AVM.* Select Window > Preferences from the main menu bar. Then open Progress OpenEdge > Shared OpenEdge AVM > AppBuilder from the tree view. Also, be aware that settings changed on the Shared OpenEdge AVM/AppBuilder preference page affect all projects that use the shared AVM.

The following controls are available:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable Dynamics</td>
<td>Enable the AppBuilder for Progress Dynamics application development in Progress Developer Studio for OpenEdge.</td>
</tr>
<tr>
<td>Configuration File</td>
<td>Specify the path of the XML configuration file where Dynamics session types are defined.</td>
</tr>
<tr>
<td>Session Type</td>
<td>Specify a name of the session type.</td>
</tr>
<tr>
<td>Qualify database fields with a database name</td>
<td>Adds a database name to all field references within the generated code.</td>
</tr>
<tr>
<td>Default to suppressing database VIEW-AS phrases</td>
<td>Suppress the Data Dictionary VIEW-AS phrases that specify a certain field to be viewed as a specific widget.</td>
</tr>
<tr>
<td>Character Terminal Colors</td>
<td>Opens the Character Terminal Simulator Colors dialog which allows you to customize the background and foreground colors of the dynamics objects and frame in the design window.</td>
</tr>
</tbody>
</table>

**Note:** For information about Progress Dynamics configuration and session types, see OpenEdge Development: Progress Dynamics Administration. You can access this and other Progress Dynamics manuals in the OpenEdge Product Documentation section of the Progress Communities Web site.
Assemblies properties page (Windows only)

The Assemblies properties page allows you to specify the location and to modify the content of the assemblies.xml file. The assemblies.xml file is a list of the assemblies where .NET classes are stored and is referenced at compile time and run time. It is automatically created for your Progress Developer Studio for OpenEdge project when you place a control on a form or manually add an assembly to the project.

To access the Assemblies page, right-click an OpenEdge project name in the Project Explorer view and choose Properties from the context menu. Expand the Progress OpenEdge node in the tree view, and then click the Assemblies node.

**Note:** If you chose the shared AVM for the currently selected project, the settings on this property page will appear dimmed. They cannot be changed on this page, and they may also be inaccurate. You must go to the Assemblies page under the Shared OpenEdge AVM preference page to view or to change these settings. Select Window > Preferences from the main menu bar. Then open Progress OpenEdge > Shared OpenEdge AVM > Assemblies from the tree view. Also, be aware that settings changed on the Shared OpenEdge AVM/Assemblies preference page affect all projects that use the shared AVM.

The following controls are available:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use default location</td>
<td>Reference the assemblies.xml file in the default location, which is the top level of the project's folder.</td>
</tr>
<tr>
<td>Workspace/File System</td>
<td>If the assemblies.xml file is not in the default location, specify some other location by navigating in the workspace or in the file system. The selection appears in the box above the controls.</td>
</tr>
<tr>
<td>assemblies.xml file viewer</td>
<td>Shows the information in the assemblies.xml file, which includes the assembly names, plus version, culture, encryption, and processor information.</td>
</tr>
<tr>
<td>Add</td>
<td>Open the Add Assembly References dialog, where you can add both global and local assemblies to the assemblies.xml file.</td>
</tr>
</tbody>
</table>
Change the order in which the assembly selected in the viewer is referenced. This is useful, for example, if you need to establish the precedence a new assembly containing patches has over the original assembly.

Remove the assembly selected in the viewer from the assemblies.xml file. When you delete a control with the Visual Designer, the assembly reference is not automatically removed from the assemblies.xml file. Although unused assembly references do no harm, you can delete them with this control.

See also
Configuration variables on page 84
Progress OpenEdge properties page on page 112
Build settings on page 116
Custom settings on page 119
Database Connections settings on page 120
PROPATH settings on page 122

Build properties page

The Build properties page consists of two tabs, Build and Source tab.

To access the Build and Source properties page, right-click an OpenEdge project name in the Project Explorer view and choose Properties from the context menu. Expand the Progress OpenEdge node in the tree view, and then click the Build node.

The Build properties page allows you to view and adjust options related to compiling ABL source code and saving r-code.

The following controls are available on the Build tab:

| Build only when source files are newer than r-code | Compile only those source files for which no up-to-date .r files are present. This option can save time when compiled code already exists. |
| **Project build destination** | The directory that will contain saved .r files. If this field is blank, .r files are saved in the same directory as the source files. The OpenEdge compiler calculates the destination by appending the project relative path of the source to the path specified in the Build destination box. If, for example, the source file is in:
C:\workspace\myproject\src\subdir1
and you specify the build destination as:
C:\workspace\rcode
the .r files end up in:
C:\workspace\rcode\src\subdir1
When you compile a .cls file, the directory structure below the path specified in the Build destination box matches the internal package name declared in the .cls file.

**Note:** The settings on the Source tab of the PROPATH properties page override the build destination specified here. Also note that the Source tab allows you to set multiple source/destination folders. Whereas, the build destination setting on the Build properties page only allows you to specify a single destination folder for all the .r files in the project. |
| **Static file destination** | The directory, relative to the project root folder, to which static files are copied. This setting is optional. |
| **Generate Sonic invocation files on build** | Automatically generate Sonic .esboe files whenever a build occurs for the project. |
| **Generate BPM invocation files** | Automatically generate BPM .bizoe files whenever a build occurs for the project. Generated .bizoe files are stored in the location specified in the File Destinations properties page. |
| **Generate Rest invocation files** | Automatically generate Data Service Catalog .json files whenever a build occurs for the project. For more information, see . |
| **Configure Workspace settings** | Use this page to specify the file extensions associated with compilable ABL files. See also Editor Build options . |
| **Compilable File Extensions** | A comma separated list of the file type extensions that the OpenEdge Editor and SpeedScript can compile. |

The Source tab allows you to specify the directories in the project where source code resides, the environment (TTY or GUI) of the source code, and the build destination for r-code. You can specify multiple sets of source/destination folders.

The following fields and controls are available on the Source tab:
## Source directory viewer
Displays a list of folders in the selected project that are designated as locations of source files under the **ROOT** node.

When you expand a directory in the list, you see four nodes: Environment (the runtime environment), Build Destination (where r-code is created), **Excluded** (lists the resources excluded from the build), and **Platform** node.

You can specify a unique environment and build destination for each source directory in your project. For projects prior to 11.0, by default a ROOT node is added on importing the project.

In releases prior to 11.0, the **Source** tab contained no entries by default. See also opening older version project with Progress Developer Studio for OpenEdge 11.0.

**Note:** Source files that are not in one of the designated source file folders shown in this view will still be built. The destination folder for the r-code will be whatever is specified as a build destination on the **Build properties** page. If there is no build destination specified, the r-code destination is the same folder that contains the source file.

---

## Add Folder
Displays a list of the folders in your project that you can designate as containers of the project's source code. It includes the root folder and any sub-folders that are resources in your project structure.

When you select a folder from the list, it appears in the Source directory viewer.

## Remove
Allows you to remove the selected folder as a container for source code. This does not remove the folder from the file system, neither does it remove the folder as a project resource.

If **Excluded** node is selected, **Remove** allows deleting the excluded entries from this node so that they are included in the build. See also including resources in the build.

## Edit
- If a source directory is selected, **Edit** allows you to change to a different source folder in the project.
- If **Environment** is selected, **Edit** allows you to designate the runtime environment (GUI or TTY) of your code. By default, the environment is all, which signifies that your code is intended to run in both environments.
- If **Build Destination** is selected, **Edit** allows you to designate where to put the r-code in the file system.
- If Excluded node is selected, **Edit** allows you to modify the excluded entries (exclude or include in the build).

## Move Up
Changes the build order. You can move the resources or projects up to the top in the build process, so that they are built first.

## Move Down
Changes the build order. You can move down the resources or projects to the bottom in the build process, so that they are built last.
See also

Setting project properties on page 98
Progress OpenEdge properties page on page 112
AppBuilder properties page Windows only on page 114
Assemblies properties page Windows only on page 115
Custom settings on page 119
Database Connections settings on page 120
Modules properties page on page 121
PROPATH settings on page 122
Configuration variables on page 84

Configuration Variables properties page

The Configuration Variables properties page allows you to view and define custom configuration variables, to make a project retain its settings when imported into another workspace, and to save settings in a named profile.

To access the Configuration Variables properties page, right-click an OpenEdge project name in the Project Explorer view and choose Properties from the context menu. Expand the Progress OpenEdge node in the tree view, and then click the Configuration Variables node.

The following controls are available:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{APP}</td>
<td>An optional variable (only valid in project properties pages) that represents the root directory for your application, and is used in path statements. Enter the appropriate path.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This is a predefined variable and cannot be deleted.</td>
</tr>
<tr>
<td>{DB}</td>
<td>An optional variable (only valid in project properties pages) that represents the root directory for your database, and is used in path statements. Enter the appropriate path.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This is a predefined variable and cannot be deleted.</td>
</tr>
<tr>
<td>Add</td>
<td>Defines a new configuration variable.</td>
</tr>
<tr>
<td>Edit</td>
<td>Edits an existing configuration variable.</td>
</tr>
<tr>
<td>Remove</td>
<td>Deletes the selected configuration variable.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> You cannot delete the predefined variables: {APP} and {DB}.</td>
</tr>
</tbody>
</table>

See also

Setting project properties on page 98
Chapter 2: Introducing OpenEdge Projects

- Importing project settings and resources on page 100
- Configuration variables on page 84
- Progress OpenEdge properties page on page 112
- AppBuilder properties page (Windows only) on page 114
- Build settings on page 116
- Database Connections settings on page 120
- PROPATH settings on page 122

Configuration Variables preferences

The Configuration Variables preference page provides a list of available configuration variables as defined for the workspace.

The following options are available:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Defines a new configuration variable.</td>
</tr>
<tr>
<td>Edit</td>
<td>Modifies aspects of an existing configuration variable. This button is disabled until a configuration variable is selected.</td>
</tr>
<tr>
<td>Remove</td>
<td>Deletes the selected configuration variable.</td>
</tr>
</tbody>
</table>

Database Connections properties page

The Database Connections properties page allows you to select database connections for a project. You can choose from all of the database connections available in the workspace. Progress Developer Studio for OpenEdge automatically connects to selected databases when you open the project.

To access the Database Connections page, right-click an OpenEdge project name in the Project Explorer view and choose Properties from the context menu. Expand the Progress OpenEdge node in the tree view, and then click the Database Connections node.

Note: If you chose the shared AVM for the currently selected project, the settings on this property page appear dimmed. They cannot be changed on this page, and they may also be inaccurate. You must go to the Database Connections page under the Shared OpenEdge AVM preference page to view or to change these settings. Select Window > Preferences from the main menu bar. Then open Progress OpenEdge > Shared OpenEdge AVM > Database Connections from the tree view. Also, be aware that settings changed on the Shared OpenEdge AVM/Database Connections preference page affect all projects that use the shared AVM.

The following controls are available:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure Database Connections</td>
<td>Opens up the Database Connections preference page where you can add, edit and delete, and import and export database connection profiles.</td>
</tr>
<tr>
<td>Show Selected</td>
<td>Show only the database connections that are associated with the currently selected project.</td>
</tr>
<tr>
<td>Show All</td>
<td>Show all database connections that are available in the current workspace.</td>
</tr>
</tbody>
</table>
Displays a list of available database connections, along with their associated attributes. Those connections which are currently assigned to the project are checked and displayed first in the list.

To assign a database connection to the current project, select the checkbox. When a row is checked, it is repositioned at the top of the list, just below the last checked item.

### Connection String

Displays the database startup parameters for the connection selected in the Connections viewer. Click on a connection name in the viewer to select it.

For more information about OpenEdge database startup parameters, see *OpenEdge Data Management: Database Administration*. You can find OpenEdge manuals in the Product Documentation section of the Progress Software Developer’s Network Web site.

### See also

- Setting project properties on page 98
- Progress OpenEdge properties page on page 112
- AppBuilder properties page Windows only on page 114
- Custom settings on page 119
- Database Connections settings on page 120
- PROPATH settings on page 122

### File Destinations properties page

The File Destinations properties page contains fields where you can specify the destination of Sonic ESB (.esboe), and BPM (.bizoe) invocation files. To set the automatic generation of either of these invocation files, open the Build properties page.

### See also

- Build properties page on page 116

### Modules properties page

The Modules properties page allows you to view and configure the modules for AppServer, WebSpeed, and Static Web project types.

**Note:** The Modules properties page displays only if an OpenEdge AppServer facet, an OpenEdge WebSpeed facet, or a Static Web facet is installed on the currently selected project. You install these facets when you create an AppServer project type, a WebSpeed project type, or a Static Web project type on the Create an OpenEdge Project page. Existing OpenEdge projects can be migrated to use these facets in the New OpenEdge Project wizard.
To access the Modules page, right-click the project name in the Project Explorer view and choose Properties from the context menu. Expand the Progress OpenEdge node in the tree view, and then click the Modules node.

The following controls are available:

| Module viewer | Displays a tree view of the module name and the folders that are configured for the project. |
| Modify | Allows you to change the module name that is configured for the selected project and to select the folders to be published. |

See also
Project types in Progress Developer Studio for OpenEdge on page 86

PROPATH properties page

To access the PROPATH properties page, right-click an OpenEdge project name in the Project Explorer view and choose Properties from the context menu. Expand the Progress OpenEdge node in the tree view, and then click the PROPATH node.

**Note:** If you chose the shared AVM for the currently selected project, the settings on this property page will appear dimmed. They cannot be changed on this page, and they may also be inaccurate. You must go to the PROPATH page under the Shared OpenEdge AVM preference page to view or to change these settings. Select Window > Preferences from the main menu bar. Then open Progress OpenEdge > Shared AVM > PROPATH from the tree view. Also, be aware that settings changed on the Shared OpenEdge AVM/PROPATH preference page affect all projects that use the shared AVM.

The PROPATH tab lets you control where the AVM searches for files and procedures.

The following controls are available on the PROPATH tab:

<p>| PROPATH viewer | Displays PROPATH settings in a tree view format. You can expand nodes to see more detail about a particular setting. It usually opens with a default list of typical settings, which you can modify with the controls on the left of the page. The settings should include all locations that might contain files needed for your application, and should be arranged in an appropriate search order. A PROPATH entry can be either a directory or a procedure library. By default all PROPATH entries are part of the OpenEdge tooling. To exclude a PROPATH entry from OpenEdge tooling, select the Exlude from OpenEdge tooling node under the PROPATH entry that you want to exclude. The Exlude from OpenEdge tooling node appears as a child node for each PROPATH entry. |
| Move Up, Move Down | Changes the position in the list of a current entry, and thus changes the search order. |
| Remove | Deletes an entry from the list. |
| Edit | Changes the value of the current entry. |
| Add Procedure Library | Adds a procedure library from the selected project. |</p>
<table>
<thead>
<tr>
<th>Add External Library</th>
<th>Adds a procedure library from the file system.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Workspace Directory</td>
<td>Adds a directory from the current workspace.</td>
</tr>
<tr>
<td>Add External Directory</td>
<td>Adds a directory from the file system.</td>
</tr>
<tr>
<td>Add Standard Paths</td>
<td>Displays a list of common locations that you can choose to add to your PROPATH setting.</td>
</tr>
</tbody>
</table>

**See also**

- Setting project properties on page 98
- Adding resources to a project on page 95
- Progress OpenEdge properties page on page 112
- AppBuilder properties page (Windows only) on page 114
- Assemblies properties page (Windows only) on page 115
- Build settings on page 116
- Custom settings on page 119
- Database Connections settings on page 120
- Modules properties page on page 121
- Configuration variables on page 84

**SpeedScript properties page**

The SpeedScript properties page applies to WebSpeed projects.

The option **Generate W file for WebSpeed** is selected by default in order to enable the automatic generation of the .w files that are necessary to debug SpeedScript.

The **SpeedScript editor e4gl code generation** field allows you to specify the destination of generated .w files. By default, the target is SpeedScriptGen, a folder in the current project.

**Project Facets project properties page**

The Project Facets page of Project Properties allows you to view, add, and remove facets from a project. This page is available only if the selected project is a faceted project (all OpenEdge projects created in 11.0 and higher). If the project facet that you add to this page contains any configuration pages, the page displays a **Further configuration** available link. You can select this link to open the corresponding pages.

**See also**

- Project types in Progress Developer Studio for OpenEdge on page 86
- Creating a new OpenEdge project
- New OpenEdge Project wizard on page 105
Progress OpenEdge Import and Export wizards

Import Data Definitions page

The Import Data Definitions page of the Import wizard allows you to import schema changes from a data definition file into the Progress database.

**Note:** This feature is limited to Progress OpenEdge databases and does not impact other databases like Oracle, SQL, etc.

To access this page, select File > Import, expand the Progress OpenEdge node in the tree view, and select the Database Definitions node.

**Note:** You can also access this page by selecting Import DF on the Database connections preferences page.

The following controls are available on the Import Data Definitions page:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input file</td>
<td>Specifies the data definition file to be imported. Enter the path or browse to select a path and the file name in the local system.</td>
</tr>
<tr>
<td>Target database</td>
<td>Specifies the active database where the schema from the selected data definitions file is loaded.</td>
</tr>
<tr>
<td>Force Commit</td>
<td>Ensures that the data dictionary continues to load and commit changes even if it encounters errors.</td>
</tr>
<tr>
<td>Add Object Online</td>
<td>Adds new objects without obtaining an exclusive lock.</td>
</tr>
</tbody>
</table>

**Note:** You cannot import data definitions into an empty database. An error is displayed when you select an empty database. An empty database refers to a database without any AREA defined. The page prompts you when the import operation is successful and displays an error if a database connection error occurs or the file location authentication fails.

See also

Export data definitions on page 126

Import workspace database connections page

The Import workspace database connections page of the Import wizard allows you to import connection information from one workspace to another. After exporting connection information from a workspace to an XML file, you can import that XML file to another workspace.

To access this page, select File > Import, expand the Progress OpenEdge node in the tree view, and select the Database connections node.

**Note:** You can also access this page by clicking Import on the Database connections preferences page.

The following controls are available on the Import workspace database connections page:
**From database connection file**  
Specifies the database connection file to be imported.

**Note:** The database connection file is an XML file created on the Export workspace database connections page.

Type the path, select a previous path from the drop down list, or browse to select a path and file name on the file system.

**Ignore duplicate connections**  
Ignores listing of connections that already exist in your workspace.

**Database connections**  
Displays a list of the connections defined in the database connection file, along with their associated attributes.

To select database connections to be imported, select their corresponding check boxes.

**Connection string**  
Displays the database startup parameters for the connection selected in the Database Connections viewer. Click a connection name in the viewer to select it.

For more information about OpenEdge database startup parameters, see OpenEdge Data Management: Database Administration. You can find OpenEdge manuals in the Product Documentation section of the Progress Software Developer's Network Web site.

**Select All**  
Selects all database connections.

**Deselect All**  
Deselects all database connections.

---

**See also**
- Exporting Database Connections on page 102
- Importing Database Connections on page 102
- Export workspace database connections page on page 127

---

**Import shared AVM settings page**

The **Import shared AVM settings** page of the Import wizard allows you to import settings used by the shared AVM from the local file system.

To access this page, select **File > Import**, expand the Progress OpenEdge node in the tree view, and select the Shared AVM settings node.

The following controls are available on the **Import shared AVM settings** page:

**From directory**  
Specifies the source directory to import the files from.

Type the path, select a previous path from the drop down list, or browse to select a path and file name on the file system.

**Shared AVM settings**  
Displays a list of settings used by the shared AVM to be imported from the local file system.

**Select All**  
Selects all shared AVM settings.
Deselects all shared AVM settings.

Overwrite existing files without warning

Replaces the existing files without a warning message.
If the specified file already exists in the file system, you will be prompted to overwrite the file. Select this option if you do not want to be prompted.

See also
Export shared AVM settings page on page 128

Import Workspace OpenEdge Explorer Connections page

The Import Workspace OpenEdge Explorer Connections page of the Import wizard allows you to import connection information from one workspace to another. After exporting connection information from a workspace to an XML file, you can import that XML file to another workspace.

To access this page expand the Progress OpenEdge > Server > OpenEdge Explorer Connections node in the Preferences page, and then click Import.

The following controls are available on the import Explorer connections page:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>From OpenEdge Explorer connection file</td>
<td>Specifies the Explorer connection file to be imported. Enter the path, select a previous path from the drop down list, or browse to select a path and file name on the file system. Note: The explorer connection file is an XML file created on the Export workspace database connections page.</td>
</tr>
<tr>
<td>OpenEdge Explorer connections</td>
<td>Displays a list of the connections defined in the Explorer connection file, along with their associated attributes.</td>
</tr>
<tr>
<td>Select All</td>
<td>Selects all OpenEdge Explorer connections.</td>
</tr>
<tr>
<td>Deselect All</td>
<td>Deselects all OpenEdge Explorer connections.</td>
</tr>
<tr>
<td>Ignore duplicate profiles</td>
<td>Ignores listing of connections that already exist in your workspace.</td>
</tr>
</tbody>
</table>

See also
Export Workspace OpenEdge Explorer Connections page on page 129

Export Data Definitions page

The Export Data Definitions page of the Export wizard allows you to export schema changes to a data definition file which can be used by other OpenEdge developers.

Note: This feature is limited to Progress OpenEdge databases and does not impact other databases like Oracle, SQL, etc.
To access this page, select **File > Export**, expand the Progress OpenEdge node in the tree view, and select the Database Definitions node.

**Note:** You can also access this page by selecting Export DF on the Database connections preferences page.

The following controls are available on the **Export Data Definitions** page:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source database</strong></td>
<td>Specifies the active database to which the data definitions are exported. You can either choose from the drop-down list of active databases or select the location and file in the local system.</td>
</tr>
<tr>
<td><strong>Select Tables</strong></td>
<td>Opens the <strong>Select Tables</strong> dialog in which you can select one or more tables for the source database. All the tables are selected by default.</td>
</tr>
<tr>
<td><strong>Create Incremental DF file</strong></td>
<td>Exports the incremental changes between two selected databases to the local system.</td>
</tr>
<tr>
<td><strong>Comparison Database</strong></td>
<td>Specifies the database that is compared to the source database and exports the incremental changes between the two.</td>
</tr>
</tbody>
</table>

**Note:** The **Select Tables** option is enabled when you select a database in the **Source** database field.

**Note:** The **Comparison Database** field is enabled when you select the **Create Incremental DF file** check box.

**Note:** The page prompts you when the export operation is successful and displays an error if a database connection error occurs or the file location authentication fails.

**See also**
Import data definitions page on page 124

**Export workspace database connections page**

The **Export workspace database connections** page of the **Export** wizard allows you to export connection information from a workspace to an XML file.

To access this page, select **File > Export**, expand the Progress OpenEdge node in the tree view, and select the Database connections node.

**Note:** You can also access this page by clicking **Export** on the **Database connections preferences** page.

The following controls are available on the **Export workspace database connections** page:
### Database connections

Displays a list of the configured database connections, along with their associated attributes.

To select the database connections to be exported to the XML file, select their corresponding check boxes.

### Connection string

Displays the database startup parameters for the connection selected in the Database Connections viewer. Click a connection name in the viewer to select it.

For more information about OpenEdge database startup parameters, see OpenEdge Data Management: Database Administration. You can find OpenEdge manuals in the Product Documentation section of the Progress Software Developer’s Network Web site.

### Select All

Selects all database connections.

### Deselect All

Deselects all database connections.

### To database connection file

Specifies the target location for the exported XML file.

Type the path, select a previous path from the drop down list, or browse to select a path and file name on the file system.

### Overwrite existing files without warning

Replaces the existing database connections files without a warning message.

If the specified database connections file already exists in the file system, you will be prompted to overwrite the file. Select this option if you do not want to be prompted.

### Exclude user ID and password

Prevents user ID or password information from being included in the exported XML file.

Select this option if you are exporting connection information to some other user.

### See also

- Exporting database connections on page 102
- Importing database connections on page 102
- Import workspace database connections page on page 124

### Export shared AVM settings page

The Export shared AVM settings page of the Export wizard allows you to export settings used by the shared AVM to the local file system.

To access this page, select File > Export, expand the Progress OpenEdge node in the tree view, and select the Shared AVM settings node.

The following controls are available on the Export shared AVM settings page:

<table>
<thead>
<tr>
<th>Shared AVM settings</th>
<th>Displays a list of settings used by the shared AVM to be exported to the local file system.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select All</td>
<td>Selects all shared AVM settings.</td>
</tr>
<tr>
<td>Deselect All</td>
<td>Deselects all shared AVM settings.</td>
</tr>
</tbody>
</table>
To directory

Specifies the destination directory for the exported files. Type the path, select a previous path from the drop down list, or browse to select a path and file name on the file system.

Overwrite existing files without warning

Replaces the existing files without a warning message. If the specified file already exists in the file system, you will be prompted to overwrite the file. Select this option if you do not want to be prompted.

See also

Import shared AVM settings page on page 125

Export Workspace OpenEdge Explorer Connections page

The Export Workspace OpenEdge Explorer Connections page of the Export wizard allows you to export Explorer connection information from a workspace to an XML file.

To access this page expand the Progress OpenEdge > Server > OpenEdge Explorer Connections node on the Preferences page, and then click Export.

The following controls are available on the the export Explorer connections page:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenEdge Explorer connections</td>
<td>Displays a list of the configured Openedge Explorer connections, along with their associated attributes.</td>
</tr>
<tr>
<td>Select All</td>
<td>Selects all OpenEdge Explorer connections.</td>
</tr>
<tr>
<td>Deselect All</td>
<td>Deselects all OpenEdge Explorer connections.</td>
</tr>
<tr>
<td>To OpenEdge Explorer connection file</td>
<td>Specifies the target location for the exported XML file. Enter a path, select a previous path from the drop down list, or browse to select a path and file name on the file system.</td>
</tr>
<tr>
<td>Overwrite existing files without warning</td>
<td>Replaces the existing OpenEdge Explorer connections file without a warning message. If the specified OpenEdge Explorer connections file already exists in the file system, you will be prompted to overwrite the file. Select this option if you do not want to be prompted</td>
</tr>
<tr>
<td>Exclude user ID and password</td>
<td>Prevents user ID or password information from being included in the exported XML file. Select this option if you are exporting connection information to some other user.</td>
</tr>
</tbody>
</table>

See also

Import Workspace OpenEdge Explorer Connections page on page 126
Build and Compile options

The following tables summarize the Build and Compile options available with Progress Developer Studio for OpenEdge. Some of these options are provided by the Eclipse framework, while others are a part of the Progress Developer Studio for OpenEdge's functionality. The tables also list the help topics associated with each option that you can refer to, for detailed information.

Options provided by Progress Developer Studio for OpenEdge:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build Working Set Automatically</td>
<td>Allows you to automatically build a working set.</td>
<td>Building a working set automatically</td>
</tr>
<tr>
<td></td>
<td>To access this option, select Project &gt; Build Working Set Automatically.</td>
<td></td>
</tr>
<tr>
<td><strong>Note:</strong> The Build Working Set Automatically option is enabled only when the automatic build preference (at the project level) is disabled (ensure that Project menu &gt; Build Automatically is not selected).</td>
<td></td>
<td>Build Working Set Automatically</td>
</tr>
<tr>
<td>Clear OpenEdge Compile Errors</td>
<td>Clears any error messages for a particular file from the Problems view or the marker bar.</td>
<td>Clear OpenEdge Compile Errors</td>
</tr>
<tr>
<td></td>
<td>To use this option, select Project &gt; Clear OpenEdge Compile &gt; Errors or select Progress OpenEdge &gt; Clear OpenEdge Compile &gt; Errors from the ABL Editor context (right-click) menu.</td>
<td></td>
</tr>
<tr>
<td>Save automatically before build</td>
<td>Saves all the modified resources in the Workbench before a manual build.</td>
<td>Save automatically before build</td>
</tr>
<tr>
<td></td>
<td>To access this option, select Windows &gt; Preferences &gt; General &gt; Workspace.</td>
<td></td>
</tr>
<tr>
<td>Compilable file extensions</td>
<td>Specifies the file extensions defined in the workspace for file types that the OpenEdge Editor and SpeedScript can compile.</td>
<td>Compilable file extensions</td>
</tr>
<tr>
<td></td>
<td>To access this option select Windows &gt; Preferences &gt; Progress OpenEdge &gt; Editor &gt; Build.</td>
<td></td>
</tr>
<tr>
<td>Compile on save if required</td>
<td>Compiles ABL source files each time you save them.</td>
<td>Compile on save if required</td>
</tr>
<tr>
<td></td>
<td>To access this option select Windows &gt; Preferences &gt; Progress OpenEdge &gt; Editor &gt; Build.</td>
<td></td>
</tr>
<tr>
<td>Exclude from Build</td>
<td>Excludes compilable resources when you build your project. You can exclude files, folders, or both from the build.</td>
<td>Exclude from Build</td>
</tr>
<tr>
<td></td>
<td>To access this option select the Progress OpenEdge menu that is available from the ABL Editor context (right-click) menu.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>You can also exclude resources using Exclusion Pattern Selection and Add Resource Exclusions. See Excluding resources using Exclusion Pattern Selection and Excluding resources using Add Resource Exclusions for more information.</td>
<td></td>
</tr>
</tbody>
</table>

Chapter 2: Introducing OpenEdge Projects
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Include in Build</td>
<td>Includes the excluded resources to the build. This option is only available for the resources that were excluded from the build. To access this option select the Progress OpenEdge menu that is available from the ABL Editor context (right-click) menu. You can also include resources using the Source tab of the Build properties page. See Including resources using the Source tab for more information.</td>
</tr>
<tr>
<td>Compile</td>
<td>Allows you to explicitly compile your current file. To access this option, select Compile from either the Source menu or the ABL Editor context (right-click) menu.</td>
</tr>
<tr>
<td>Save r-code</td>
<td>Saves .r files when you build the code. When this option is enabled, OpenEdge saves compiled .r files in the directory you specify in the Build destination field; or, if that field is blank, in the same directory as the source file. To access this option, select Properties &gt; Progress OpenEdge &gt; Build.</td>
</tr>
<tr>
<td>Build only when source files are newer than r-code</td>
<td>Compiles only those source files that do not have the latest r-code. This option can save time when compiled code already exists. To access this option, select Properties &gt; Progress OpenEdge &gt; Build.</td>
</tr>
<tr>
<td>Project build destination</td>
<td>Specifies the directory that will contain the r-code. If this field is blank, the r-code is saved in the same directory as the source files. To access this option, select Properties &gt; Progress OpenEdge &gt; Build.</td>
</tr>
<tr>
<td>Pre-compile callback routine</td>
<td>Specifies a procedure that runs prior to compilation. To access this option, select Properties &gt; Progress OpenEdge &gt; Build.</td>
</tr>
<tr>
<td>Source tab</td>
<td>Allows you to specify the directories in the project where source code resides, the environment (TTY or GUI) of the source code, and the build destination for r-code. To access this option, select Properties &gt; Progress OpenEdge &gt; Build.</td>
</tr>
<tr>
<td>Compile on publish</td>
<td>Allows you to compile code when you publish to a remote AppServer or WebSpeed server.</td>
</tr>
<tr>
<td>Compile options</td>
<td>Allows you to add any compile options, such as startup or other parameters.</td>
</tr>
</tbody>
</table>
Options provided by the Eclipse framework:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build All</td>
<td>Performs an incremental build on all the projects in the Workbench. It builds (compiles) all the resources in the Workbench that are affected by any resource changes since the last incremental build. To use this option, select Project &gt; Build All. To build all resources, even those that have not changed since the last build, select Project &gt; Clean before building the project. Note: The Build All option is only available if the Build Automatically menu option is not selected. You can access this option from the Project menu or from the General &gt; Workspace preference page.</td>
</tr>
<tr>
<td>Build Project</td>
<td>Builds the selected projects in the workspace. To use this option, you can: • Select the project that you want to build and then select Project &gt; Build Project • Select Build Project from the project context (right-click) menu Note: The Build Project option is only available if the Build Automatically menu option is not selected. You can access this option from the Project menu or from the General &gt; Workspace preference page.</td>
</tr>
<tr>
<td>Build Working Set</td>
<td>Builds (compiles) all the resources in the working set that are affected by any resource changes since the last build. It also allows you to create a new working set by using Select Working Set. To access this option select Project &gt; Build Working Set. Note: The Build Working Set option is only available if the Build Automatically menu option is not selected. You can access this option from the Project menu or from the General &gt; Workspace preference page.</td>
</tr>
<tr>
<td>Build Automatically</td>
<td>Allows you to toggle the auto build preference. When checked, builds are invoked automatically as resources are saved. Automatic builds are always incremental and always operate over the entire workspace. To access this option, select Project &gt; Build Automatically. You can also configure your preferences (Window &gt; Preferences &gt; General &gt; Workspace) to perform builds automatically on resource modifications.</td>
</tr>
</tbody>
</table>
### Build Order

Build Order allows you to explicitly define the order in which projects are built. Alternatively, you can let the platform compute the build order by interpreting project references as prerequisite relationships. The build order is applied for building the entire workspace or for a group of projects.

To access the build order options, select **Windows > Preferences > General > Workspace > Build Order**.

The following build order options are provided:

- **Up** and **Down**: Manually changes the build order.
- **Add Project** and **Remove Project**: Add and removes projects from the build order.
- **Use default build order**: Allows the platform to compute the order in which the projects are built. Turning off this option enables access to the projects list, the order of which can be manipulated.
- **Max iterations when building with cycles**: Allows you to specify the maximum number of times the workbench will attempt to iterate over the build order before giving up.

You can also use the **Source tab** of the **Build properties** page to change the build order, add, and remove projects from the build order.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| **Build Order**                             | Allows you to explicitly define the order in which projects are built. Alternatively, you can let the platform compute the build order by interpreting project references as prerequisite relationships. The build order is applied for building the entire workspace or for a group of projects. To access the build order options, select **Windows > Preferences > General > Workspace > Build Order**. The following build order options are provided:  
  - **Up** and **Down**: Manually changes the build order.  
  - **Add Project** and **Remove Project**: Add and removes projects from the build order.  
  - **Use default build order**: Allows the platform to compute the order in which the projects are built. Turning off this option enables access to the projects list, the order of which can be manipulated.  
  - **Max iterations when building with cycles**: Allows you to specify the maximum number of times the workbench will attempt to iterate over the build order before giving up. You can also use the **Source tab** of the **Build properties** page to change the build order, add, and remove projects from the build order. |
| **Wait for ongoing build to complete before launching** | Controls whether to wait for the ongoing build to complete before launching other resources. To access this option select **Windows > Preferences > Run/Debug > Launching**. |
| **Build (if required) before launching**     | Allows you to manually build the project before launching, when the Build Automatically option is not selected. To access this option select **Windows > Preferences > Run/Debug > Launching**. |
AppServer support in Progress Developer Studio for OpenEdge

The AppServer™ is the core of OpenEdge application and integration services and is the engine for running ABL business logic that can be made available to application clients as application services. Essentially, the AppServer is an ABL runtime client that has no user interface but provides a means for client applications to call its ABL procedures and user-defined functions. Like most OpenEdge server products, it relies on the Unified Broker framework for configuration and administration.

The following enhancements for Progress Developer Studio for OpenEdge (formerly known as Progress OpenEdge Architect) are now available for the AppServer:

- **Support for remote publishing** - Using OpenEdge Explorer, you can publish an application to a remote AppServer on a machine that is properly configured.

- **Additional project functionality** - You can associate a project with both AppServer and WebSpeed if the project contains the required facet.

The inclusion of runtime and project support for AppServers in OpenEdge Architect relies on the Eclipse Web Standard Tools (WST).

With AppServer support, you are able to:

- Create and publish an application to a local or remote AppServer, which allows you to test the application in an environment outside of the Progress Developer Studio for OpenEdge workspace.

- Use OpenEdge Explorer to create and manage AppServer brokers.

You can also use OpenEdge Explorer to publish to a remote machine (minimum version OpenEdge 11.0 required). Additionally, you can access OpenEdge Explorer from within Progress Developer Studio for OpenEdge.

- Create a custom publish directory for a local AppServer broker.
Optionally compile as you publish, when publishing remotely.

Create custom project types.

Monitor all brokers from a single view.

With this release of Progress OpenEdge, Progress Explorer is no longer available.

**Debugging support**

Progress Developer Studio for OpenEdge now supports debugging of remote and local AppServer instances. For more information, see *AppServer and WebSpeed debugging*.

For details, see the following topics:

- Concepts
- Tasks
- Reference

**Concepts**

**AppServer support in Progress Developer Studio for OpenEdge**

The AppServer™ is the core of OpenEdge application and integration services and is the engine for running ABL business logic that can be made available to application clients as application services. Essentially, the AppServer is an ABL runtime client that has no user interface but provides a means for client applications to call its ABL procedures and user-defined functions. Like most OpenEdge server products, it relies on the Unified Broker framework for configuration and administration.

The following enhancements for Progress Developer Studio for OpenEdge (formerly known as Progress OpenEdge Architect) are now available for the AppServer:

- **Support for remote publishing** - Using OpenEdge Explorer, you can publish an application to a remote AppServer on a machine that is properly configured.

- **Additional project functionality** - You can associate a project with both AppServer and WebSpeed if the project contains the required facet.

The inclusion of runtime and project support for AppServers in OpenEdge Architect relies on the Eclipse Web Standard Tools (WST).

With AppServer support, you are able to:

- Create and publish an application to a local or remote AppServer, which allows you to test the application in an environment outside of the Progress Developer Studio for OpenEdge workspace.

- Use OpenEdge Explorer to create and manage AppServer brokers.

You can also use OpenEdge Explorer to publish to a remote machine (minimum version OpenEdge 11.0 required). Additionally, you can access OpenEdge Explorer from within Progress Developer Studio for OpenEdge.

- Create a custom publish directory for a local AppServer broker.

- Optionally compile as you publish, when publishing remotely.
• Create custom project types.
• Monitor all brokers from a single view.

With this release of Progress OpenEdge, Progress Explorer is no longer available.

Debugging support
Progress Developer Studio for OpenEdge now supports debugging of remote and local AppServer instances.
For more information, see AppServer and WebSpeed debugging.

See also
The AppServer in the OpenEdge Server perspective on page 137
Project support for AppServers on page 139
Defining an AppServer server on page 142
Working with AppServer connections and agents on page 147
Working with AppServer projects on page 150
Publishing AppServer code to a server for testing on page 154
Progress OpenEdge Server preferences on page 162
Progress OpenEdge Server Monitor view on page 156

The AppServer in the OpenEdge Server perspective

The AppServer appears in the OpenEdge Server perspective, which includes the various views that you use to perform related activities in Progress Developer Studio for OpenEdge. The perspective is based on the OpenEdge Editor perspective but also includes the Servers view and the Progress OpenEdge Server Monitor view.

By default, the OpenEdge Server perspective includes the following views:

• Project Explorer - A hierarchical view of the projects and resources in a workspace. The view is an adaptation of the Workbench Project Explorer view with some OpenEdge-specific menu options.

• Outline - A list of the structural elements of the file that is currently open in the ABL Editor or an XML Editor. The view is the Workbench Outline view customized for working with ABL.

• DB Structure - An OpenEdge view that displays the schema of connected databases. You can drag and drop schema elements (tables, fields, and others) from this view into a file that is open in the ABL Editor.

• Console - A Workbench view that displays the text output from commands (runtime startup, for example) similar to the output in a command window. For the AppServer, the Console view displays the broker and server .log files.

• Problems - A log of the errors, warnings, and other information associated with a file that has been opened in the ABL editor. The view is the standard Workbench Problems view.

• Tasks - A To-Do list. The view is the standard Workbench Tasks view.

• Properties - The properties of the object currently selected in the Workbench.

• Servers - A list of all defined servers, including AppServers. This list appears if the AdminServer is running; otherwise, the view is blank.

• Progress OpenEdge Server Monitor - A status view for a server's broker and its agents, including those for AppServers.
See also
Publishing on page 138
Project support for AppServers on page 139
Accessing the AppServer in the OpenEdge Server perspective on page 141
Working with AppServer connections and agents on page 147
Viewing AppServer broker status, agent status and agent state on page 149
Working with AppServer projects on page 150
Publishing AppServer code to a server for testing on page 154
AppServer-related views, pages, and wizards on page 155

AppServer launch configuration

A launch configuration is a group of settings that govern an instance of the OpenEdge AVM. You can define the settings for one or more launch configurations for each AppServer instance.

It is not necessary for you to manually create a new launch configuration, as the WTP server framework creates one automatically-provided the AdminServer is running-if one does not already exist for an AppServer definition. You can, however, modify the existing one or define an alternate one that you can use to start the AppServer.

For more details, see Managing launch configurations in the Running and Debugging ABL Programs help.

Publishing

You can use the Eclipse publishing functionality to copy files from an AppServer project module in the Workspace environment to a local or remote server. This allows the server to find and use them as you test your application code.

The module definition specifies which objects must be published. For an AppServer project module, the objects come from the set of directories defined by the module. Therefore, a publish action might proceed differently depending on the location of the AppServer and its publish directory.

Events that require publishing to the server

The following events require that you publish to the server:

• Adding or changing a file in a module
• Adding a directory or subdirectory to a module
• Removing a file from a module
• Renaming an existing directory in a module
• Changing the server publish directory, which includes either changing from the working directory of the server to a specific directory or changing the selected publish directory to a different value
• Modifying the build path or PROPATH of a module's project
• Adding a module to a server
• Removing a module from a server

Publishing to a local server

When you initiate a publish to a local server, the system can activate a script to copy the code from the project's directory. The script publishes only the code that has changed since the last publish event.
You can review the server status in the **Servers** view to see whether code has been published, or if there have been changes and a publish is required. For AppServers that reside on your local machine, the server need not be running for a publish to occur.

**Note:** If the publish directory points to a location in the workspace such as a project directory, the publish is not performed. This avoids a situation in which you might corrupt source code. In a case such as this, the publish of any modules to the server does not complete, and an error message informs you of this fact.

### Publishing to a remote server

You can publish your AppServer code to a remote server by using OpenEdge Explorer (or OpenEdge Management if you have a license). You must configure the remote machine to work with the local installation of OpenEdge Explorer.

For AppServers that reside on a remote machine, the server need not be running for a publish to occur. If you select the option to compile on publish, however, the broker must be running so that source compilation can be done using the broker’s AVM.

**Note:** For more information, see **Publishing AppServer code to a remote server.**

### Removing published content from the server

When a file is removed from a module, the corresponding file must also be removed from the server. A file might have been removed from a module by either of the following actions:

- The file or folder was a member of the module and was physically deleted.
- A directory was removed from the module content definition.

In either case, when the module is next published to the server, the publish mechanism must make sure that the file is no longer present on the server. This applies to both source files and r-code.

You can set a preference to retain the removed module content in the server publish directory. (The default for this preference is to remove deleted files and folders.)

When a module is removed from the server, the publish mechanism must ensure that the module content is removed from the server. The removal of resources occurs on the next publish of the server (and could take some time if the module is large).

For additional information, see the details about publishing applications to the server in the *Web Tools Platform User Guide.*

**See also**

- Publishing AppServer code to a server for testing on page 154
- Using the Clean option before republishing on page 154
- Configuring AppServer project module properties on page 152
- Server Editor on page 160
- AppServer-related preferences on page 161

### Project support for AppServers

Most operations that you perform with Progress Developer Studio for OpenEdge take place within the context of a project. You can create multiple projects and maintain them concurrently in your workspace.

You can create an AppServer project that contains additional natures and an AppServer module. For details, see **Creating an AppServer project** on page 150.
See also

*Working with AppServer projects* on page 150

**Terminology related to AppServer support**

The following terms describe various components that apply to AppServer support in Progress Developer Studio for OpenEdge:

<table>
<thead>
<tr>
<th>Server</th>
<th>An instance of a server runtime. You can define many servers for a given runtime. For AppServer support in Progress Developer Studio for OpenEdge, a server is used to connect to OpenEdge Explorer and provides connection information, broker name, and publishing options for a particular AppServer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server configuration</td>
<td>A collection of files used to configure a running server. Typically, the configuration consists of a set of port numbers and other properties.</td>
</tr>
<tr>
<td>Server Runtimes (or runtimes)</td>
<td>The software application that executes as a server platform. For OpenEdge, this refers to the location of the OpenEdge installation that is used to communicate with OpenEdge Explorer and provide AppServer functionality.</td>
</tr>
<tr>
<td>Modules</td>
<td>A collection of files used to organize content that is being developed for a server. A module usually consists of a hierarchy of files that reside in the Eclipse workspace. Modules also have a many-to-many relationship with other modules, and the file hierarchy for modules does not indicate any specific module containment. An OpenEdge AppServer project is allowed one module, and an AppServer module is required for publishing code to a server.</td>
</tr>
<tr>
<td>Facets</td>
<td>A mechanism used to add project functionality that you can turn on or off. Within OpenEdge, you can use facets to add a specific module type to a project. When you add a facet to a project, the facet is used as a marker for performing certain setup requirements or for simply marking the project as a specific project type.</td>
</tr>
</tbody>
</table>

For more information, see the details in overview to the *Web Tools Platform Guide*.

See also

*The AppServer in the OpenEdge Server perspective* on page 137
*Project support for AppServers* on page 139
*Defining an AppServer server* on page 142
*Defining the OpenEdge AVM runtime* on page 145
*Working with AppServer projects* on page 150
*Server Editor* on page 160
*New Server wizard* on page 157
Tasks

This section contains general information about the following:

• Working with the AppServer in OpenEdge Architect.
• Working with AppServer connections and agents.
• Monitoring AppServer broker and agent activity.
• Creating AppServer projects and publishing application code for testing.

Accessing the OpenEdge Server perspective

To open the OpenEdge Server perspective:

1. Choose one:
   a) Click Open Perspective on the Workbench toolbar.
   b) Select Window > Open Perspective.

2. Select OpenEdge Server, if it is listed. If it is not listed, choose Other. Choose OpenEdge Server from the Open Perspective dialog, and then click OK.

3. To view the OpenEdge Server perspective if it is open but not in view, click OpenEdge Server perspective.

Note: For more information about using perspectives, toolbars, and menu bars, see the Workbench User Guide in the Eclipse help system.

See also

The AppServer in the OpenEdge Server perspective on page 137
Working with AppServer connections and agents on page 147
Viewing AppServer broker status, agent status, and agent state on page 149
Working with AppServer projects on page 150
Progress OpenEdge Server Monitor view on page 156
Server Editor on page 160

Setting Progress OpenEdge Server preferences

You can set several Progress OpenEdge Server preferences. Setting preferences allows you to establish certain aspects of AppServer project functionality.

To set the preferences:

1. Choose one:
   • From the Progress Developer Studio for OpenEdge menu bar, select Window > Preferences. The Preferences page appears. Select Progress OpenEdge > Server.
• From the Progress OpenEdge Server Monitor view, click the drop-down menu option on the view toolbar and select Preferences.

2. Set each of the following preferences:

• WebSpeed Broker URL
• Trim agents on publish
• Remove deleted files and folders on publish
• Remove all files and folders when cleaning server publish directory
• Publish empty folders
• Run/Debug — Update properties from server before starting/launching
• Publish filters
• Filter
• Server status refresh interval
• Enable Agent State tab

**Note:** You can alternatively click Restore Defaults to restore the preferences to their original settings.

3. You can also add or remove a filter as follows:

   a) To add a filter, click Add. Type the filter name in the Filter field, and click Update. The new filter appears in the Publish filters list.

   b) To remove a filter, select a filter and click Remove.

4. When you finish, click Apply.

5. Click OK.

**See also**

Publishing AppServer code to a server for testing on page 154
Accessing OpenEdge Explorer on page 147
Progress OpenEdge Server Monitor view on page 156

### Defining an AppServer server

Use the New Server wizard to define a connection to OpenEdge Explorer and select an AppServer broker to manage. Note that you cannot create or define a new AppServer broker from within Progress Developer Studio for OpenEdge. To create a new AppServer broker, use OpenEdge Explorer.

**Note:** From the New Server wizard, you can also click Download additional server adapters to download and install the WTP Server Adapter and additional development tools for a number of third-party server adapters.

To define an AppServer server:

1. From the Servers view in the OpenEdge Server perspective, right-click and choose New > Server from the Context menu. The Define a New Server dialog appears, allowing you to choose either OpenEdge AppServer or OpenEdge WebSpeed as the type of server you want to define.

2. To choose AppServer as the server type, either:
• Type **AppServer** in the **Select the server type** field, which filters the results to show only available AppServers.

• Select **Progress Software Corporation > OpenEdge AppServer** as the server type from the list provided.

**Note:** The server host name **localhost** is prefilled. The **Server name** and the **Server runtime environment** details are also listed in the dialog.

3. If you want to specify a different server runtime environment, click **Add**. The **New Server Runtime Environment** dialog appears.

4. Type the runtime environment name and the OpenEdge version number in the fields provided; then click **Finish**.

5. To set up the runtime environment, click **Configure runtime environments** and follow the steps provided in **Defining the OpenEdge AVM runtime**.

6. Click **Next**. The **Define a new AppServer broker** dialog appears, with the **Server name** field prefilled.

7. Do one of the following:
   - Select the name of the connection from the OpenEdge Explorer connection dropdown list.
   - Click **Configure**. The **Progress OpenEdge > Server > OpenEdge Explorer Connections Preferences** page opens. For information about how to configure an OpenEdge Explorer connection, see **Adding an OpenEdge Explorer connection**. To return to the Define a new AppServer broker dialog, click **OK**.

8. Select the broker from the dropdown list, or click the **Refresh** button to verify that you are selecting from all available brokers. The list of brokers includes all local and remote brokers, in alphabetical order, that are managed by the OpenEdge Explorer connection you specify. The machine name is appended to the broker name, so that you can distinguish among brokers that have the same name but reside on different hosts.

   **Note:** Remember that you cannot create a new AppServer broker in Progress Developer Studio for OpenEdge; to create a broker, use OpenEdge Explorer. When you select the broker, the server name is updated to identify the name of the broker, the name of the selected runtime, and the name of the host where the broker resides.

9. Click **Next** to define the publishing rules. You can choose a custom publish directory for a local broker. (For a broker that resides remotely, the option to choose a custom publish directory is unavailable.)

10. Choose one:
   - To publish to a custom publish directory for a local broker, select the check box and type the path to the directory in the **Publish directory** field; if you prefer, you can click **Browse** to locate the directory on the local machine. The publish directory you specify is created as a broker property in the ubroker.properties file for the local host. If you prefer not to use a custom publish directory for a local broker, do not select the check box. The broker’s working directory will be used as the publish location.
   - To publish to a directory for a remote broker, you must configure the publish directory by using Open Edge Explorer. See **Adding an OpenEdge Explorer connection** for more information.

11. Select or deselect the **Publish source code** and/or the **Publish r-code** options. (These options are selected by default.)

   **Note:** If you select only the **Publish source code** option, you can select the **Compile on publish** check box. Selecting this option means that whenever you publish source code to a server, it is compiled.
12. Click Next. The Add and Remove dialog appears, showing the available project modules.

13. To configure an available project module, either select the module in the Available list and click Add, or double-click the module in the Available list. To configure all the available project modules, click Add All.

Note: You can also remove a project module from the Configured list by either selecting the module and clicking Remove or double-clicking the module in the Configured list. You can also select more than one project module and click Remove All.

14. Click Finish.

Note: The wizard validates the connection and broker name information. The server queries the broker properties for PROPATH, startup parameters, and connection procedures and stores them with a new launch configuration associated with the server. This launch configuration is created and used as the default launch configuration when the server starts. For more information about launch configurations, see Launch configurations for running and debugging programs.

See also
Publishing on page 138
Project support for AppServers on page 139
Terminology related to AppServer support on page 140
Accessing OpenEdge Explorer on page 147
Defining the OpenEdge AVM runtime on page 145
New Server wizard on page 157

Adding an OpenEdge Explorer connection

To publish AppServer code remotely, you must have a configured OpenEdge Explorer connection.

Note: The Use secure connection option visible on this page is not implemented.

To add an OpenEdge Explorer connection:

1. Choose Window > Preferences.
2. Select Progress OpenEdge > Server > OpenEdge Explorer Connections.
3. The OpenEdge Explorer Connections page opens. Do the following:
   a) Click Add. The Add OpenEdge Explorer Connection dialog opens.
   b) Type the connection name.
   c) Type the Host name or IP address.
   d) Specify the port. (The default port is 9090.)
   e) Type the user name. (The default user name is admin.)
   f) Type the password. (The default password is admin.)
g) Provide the OpenEdge Explorer URL. (If you are using the default host name and port number, the URL is http://localhost:9090.)

h) To finish the connection setup, click Finish, or click Create Default Server and Finish. If there are no default servers available in the Servers view, a default server will be created and will appear in the Servers view.

4. Click OK to close the OpenEdge Explorer connection.

### Defining the OpenEdge AVM runtime

The OpenEdge AVM (ABL Virtual Machine) runtime defines for Eclipse the location of one or more OpenEdge runtime installations and allows you to configure which installation you want to use. You can set up multiple OpenEdge installations, or you can reuse a single installation.

The first time you run Progress Developer Studio for OpenEdge, it automatically creates two runtimes (one for the AppServer server type, and one for the WebSpeed server type) for the current installation. You can create a new runtime for the current version or for other installations where a server product is installed.

To specify the server runtime you want to use, begin with the Eclipse Server Runtimes Environment preferences. Choose a runtime from the list of existing OpenEdge runtimes on the preferences page, or create a new OpenEdge runtime.

1. To access the list of installed runtime environments:
   a) From the Progress Developer Studio for OpenEdge menu bar, select Window > Preferences. The Preferences page appears.
   b) Select Server > Runtime Environments. The Server Runtime Environments page appears, showing the list of all available OpenEdge runtimes.

2. You can now do the following:
   a) Use, modify, remove, or search for an installed OpenEdge runtime environment
   b) Add an OpenEdge installed runtime environment
   c) Add a third-party runtime environment

3. To use, modify, remove, or search for an installed OpenEdge runtime environment, from the Server Runtime Environments preferences page, do one of the following:
   • To use an existing OpenEdge runtime environment, select it and click OK. The Preferences page closes.
   • To modify an existing OpenEdge runtime environment, select it and click Edit. The Edit Server Runtime Environment dialog appears. Modify the runtime environment name and/or OpenEdge version, and click Finish. Then click OK to close the Server Runtime Environments preference page.
   • To remove an existing OpenEdge runtime environment, select it and click Remove. The runtime environment is deleted from the list. This deletion invalidates all existing servers that are relying on that runtime.
   • To reuse a previously removed runtime, recreate it (if necessary) and reassign the runtime for the server from the Server Editor.
   • To search for an existing OpenEdge runtime environment that does not appear in the list, click Search. The Search For Runtime Environments dialog appears. Locate the runtime environment directory and click OK. Then click OK to close the Preferences page.

4. To add an OpenEdge installed runtime environment, from the Server Runtime Environments preferences page, you can add an OpenEdge installed runtime environment.
a) Click **Add**. The **New Server Runtime Environment** dialog appears.

b) Type **OpenEdge** in the **Select the type of runtime environment** field to filter the results, or select **Progress Software Corporation OpenEdge AppServer** as the runtime environment from the list provided.

c) To create a new local server, select the **Create a new local server** check box.

d) Click **Next**.

e) Type the Runtime environment name and OpenEdge version in the fields provided.

f) If you chose to create a new local server in Step 3, click **Next**, define a new AppServer broker by **adding the OpenEdge Explorer connection**, and click **Next**, **define the publishing rules**, click **Finish**, and click **OK**. If you did not choose to create a new local server, click **Finish**, and click **OK**.

5. To add a third-party runtime environment, from the **Server Runtime Environments Preferences** page, you can click **Add** and then click **Download additional server adapters** to download and install the Eclipse WTP Server Adapter and additional development tools for a number of third-party server adapters.

---

**Note:** For more information, see the overview in the **Web Tools Project help**.

---

**See also**

**Server Runtime Environments preferences** on page 162

---

**Viewing or modifying AppServer properties**

By using the Server Editor, you can view or modify the AppServer properties that define the connection to OpenEdge Explorer and the AppServer broker.

To view or modify the AppServer properties:

1. From the **Servers** view in the OpenEdge Server perspective, open the **Server Editor** in either of the following ways:
   - Double-click the server name.
   - Select the server name, right-click, and select **Open**.

2. Review the AppServer properties in the following categories:
   - **General Information**
   - **Connection**
   - **Publishing**
   - **Timeouts**
   - **Publish Location**

3. Make any changes, and click **OK**.
Accessing OpenEdge Explorer

You can access OpenEdge Explorer from several different perspectives, including the OpenEdge Server. (It might be necessary for you to reset the perspectives in existing workspaces to see new view and perspective shortcuts.)

From the Progress Developer Studio for OpenEdge menu bar, select **OpenEdge > Admin > OpenEdge Explorer**. Depending on whether you have installed OpenEdge Management (which includes OpenEdge Explorer) or just OpenEdge Explorer, one of the following occurs:

- If you have OpenEdge Management installed, the login window appears. Type your user name and password, and click **OK**. The Web browser opens to the management console.
- If you have OpenEdge Explorer only, the OpenEdge Explorer connection defined in the **OpenEdge Explorer Connections** page is used to open the Web browser.

See also

Progress OpenEdge Server preferences on page 162

Working with AppServer connections and agents

See also

The AppServer in the OpenEdge Server perspective on page 137
Progress OpenEdge Server Monitor view on page 142
Configuring a JDBC driver on page 156
Server Editor on page 160

Starting and stopping the AppServer

You can start or stop the AppServer from the Servers view in the **OpenEdge Server perspective**.

1. From the **Servers** view, highlight an AppServer.

2. Open the **Server Editor** by doing either of the following:
   - Double-click the name of the server.
   - Select the server name, and right-click. Then select **Open**.

3. Provide the OpenEdge Explorer connection details in the **Connection** section of the Server Editor, or click **Configure** to add a connection.

4. Choose one:
   - To start or restart the server, select the server name in the **Servers** view and click **Start** on the view toolbar, or right-click the server name and click **Start**.
   - To start or restart the server in debug mode, either select the server in the **Servers** view and click **Start the server in debug mode** on the view toolbar, or right-click the server name and select **Debug**. In addition, you must be sure that the broker's machine is enabled for debugging.

5. For a local broker's server, enabling debugging is done automatically when the server is launched in debug mode. For a remote broker's server, you must do the following:
a) From the left pane of the management console on the remote machine, select **Resources > [container name] > OpenEdge > AppServer**, and then select the broker.

b) Click **Configuration**. In the **Advanced Features** area of the Broker properties, be sure that the **Enable debugging through the broker property** is selected.

6. To stop the server, either select the server in the **Servers** view and click **Stop** on the view toolbar, or right-click the server name and select **Stop**.

**Note:** The **Servers** view refreshes to show the state as one of the following: **Starting**, **Started**, **Stopping**, or **Stopped**. The status can also be blank (which is the default) if the state is unknown. The **Status** is also refreshed to show one of the following: **Synchronized**, **Publishing**, or **Republish**.

**See also**
- Viewing AppServer broker status, agent status, and agent state on page 149
- Progress OpenEdge Server Monitor view on page 156

**Adding AppServer agents**

When agent requests from AppServer clients are numerous, you can increase the number of agents running and available to service those requests. Likewise, if there is a reduction in the number of requests, you can reduce (trim) the number of running AppServer agents.

To add one or more AppServer agents:

1. From the **Servers** view, select a running AppServer broker and right-click. The **Context** menu appears.
2. Select **Add Agents**. The **Add Agents** dialog appears.
3. Type the number of agents you want to add in the **Count** field.
4. Click **OK**.

**See also**
- Trimming AppServer agents on page 148
- Add/Trim Agents dialog on page 161

**Trimming AppServer agents**

When there is a reduction in the number of agent requests by AppServer clients, you can trim the number of agents running and available to service those requests.

Likewise, if the number of agent requests increases, you can add AppServer agents to the process pool.

To trim one or more running AppServer agents:

1. From the **Servers** view, select an AppServer and right-click. The **Context** menu appears.
2. Choose one:
   - To trim all AppServer agents, select **Trim all Agents**.
   - To specify the number of agents you want trimmed, select **Trim Agents**. Type the number of agents you want to trim in the **Count** field, and then click **OK**.
Note: If an agent is busy, the trim request might fail for that particular agent.

See also
Adding AppServer agents on page 148

Renaming an AppServer connection

You can rename an AppServer connection that you have defined.

To rename the connection:

1. From the Servers view, select the AppServer name and right-click.
2. Select Rename. The name of the server becomes modifiable in the Servers view.
3. Type the new name of the AppServer connection, and press ENTER.

See also
Defining an AppServer server on page 142
Deleting an AppServer connection on page 149

Deleting an AppServer connection

You can delete an AppServer connection that you have defined.

To delete the connection:

1. From the Servers view, select the AppServer and right-click.
2. Select Delete.
3. Select the Delete running server(s) option. If you want to stop an active connection before you delete it, select the Stop server(s) before deleting option.
4. Click OK.

See also
Defining an AppServer server on page 142
Renaming an AppServer connection on page 149

Viewing AppServer status

You can view AppServer status by looking at:

- The status of brokers and status and state of agents
- The server and broker log files

Viewing AppServer broker status, agent status, and agent state

You can view the status of an AppServer broker and agents in the Progress OpenEdge Server Monitor view.
1. Open the Progress OpenEdge Server Monitor view in any of the following ways:
   - From the OpenEdge Server perspective, click the Progress OpenEdge Server Monitor view tab.
   - From any other perspective, select Window > Show View > Other > Server > Progress OpenEdge Server Monitor.
   - From the Servers view, right-click and select Progress OpenEdge Server Monitor from the context menu.

2. Type filter text in the available field to see a particular broker, or highlight the AppServer category and expand its contents.

3. Click a broker in the AppServer list to see a summary of status information for the broker and its agents.

4. To automatically refresh the view, you can opt to automatically refresh the Progress OpenEdge Server Monitor view. From the view, click Automatically refresh server status in the view toolbar.

5. To refresh the server status, click Refresh server status in the view toolbar.

   **Note:** You can also set the Server status refresh interval preference if you want the status to refresh automatically at an interval you specify.

---

**Displaying the AppServer broker and server log files**

You can display a broker log file or a server log file for the AppServer from the Console view. The Console view is available when you start or restart the server from within Eclipse. (If you started the server either from outside of Progress Developer Studio for OpenEdge or by a different Progress Developer Studio for OpenEdge session, the Console view is unavailable.)

You can open the Console view to display the broker log file or the server log file in any of the following ways:

   - From the Servers view, select the server or broker and right-click. Then choose Show In > Console.
   - From the Progress Developer Studio for OpenEdge menu bar, select Window > Show View > Console view, click Display Selected Console from the toolbar.
   - From the Progress Developer Studio for OpenEdge menu bar, select Window > Show View > Console and from the Console view, click Display Selected Console from the toolbar and select the broker/server log from the dropdown list.

**See also**

- Starting and stopping the AppServer on page 147
- Viewing AppServer broker status, agent status, and agent state on page 149

---

**Working with AppServer projects**

This section describes the procedures for working with AppServer projects.

**Creating an AppServer project**

You have to create an AppServer project with the modules to build an application that can be deployed to PAS for OpenEdge or an AppServer.
To create a new project:

1. Select **File > New > OpenEdge Project** from the Progress Developer Studio for OpenEdge main menu bar or the Project Explorer view context (right-click) menu. The New OpenEdge Project wizard opens and displays the **Create an OpenEdge Project page** on page 105.

2. Type a name in the **Project name** field.
   The root folder for the new project is created in your current Eclipse workspace, unless you specify a different location.

3. To choose a different location for the project, click and clear the **Use default** check box. Then either browse to, or enter the path to the desired location.

4. Select **Server** from the left section.

5. Select either **PAS for OpenEdge** or **Classic** option button to select the server type that you want to deploy the application to.

6. Select **APSV** as the transport.

7. Click **Next**.

8. Choose whether to use a project-specific AVM or a shared AVM.

9. Clear the **Use TTY for runtime** check box if you want to enable a facet for this project that substitutes another runtime environment.

10. Specify the OpenEdge AVM options and the project layout options (or retain the default settings) and click **Next**. The **Define AppServer content module** page opens.

11. Type the module name and the AppServer source folder name in the fields provided. If you have multiple projects and modules, give each module a unique name.

   **Note:** The AppServer source folder is automatically included as part of the project when you finish the project creation. The folder is also added to the PROPATH for the project as a source directory, and the default r-code destination is the same as that of the project. You can later add any files manually to this source folder that must be published to the AppServer or to PAS for OpenEdge.

12. Select one or more servers that you want to publish the AppServer module to and click **Next**.

13. Specify the PROPATH to use for the project and click **Next**.

14. Specify the database connection(s) to use for the project and click **Finish**.

   **Note:** You can add other module content resources through the **AppServer module configuration property** page after creating the project.

**See also**

- **Publishing** on page 138
- **Defining the OpenEdge AVM runtime** on page 145
- **Publishing AppServer code to a server for testing** on page 154
Configuring AppServer project module properties

The Modules configuration property page allows you to configure how a module is deployed to the server with which it is associated. You can configure properties for any new projects or for any existing OpenEdge projects that you have migrated to use facets (to support modules).

The property page allows you to identify the folders that you want published to the server, as well as the file types that you want deployed as part of the module. You can specify additional folders that are included in the current project, even linked folders, as well as remove any folders you no longer want included.

1. To access the module configuration property page, do either of the following:
   - Select the AppServer project in the Project Explorer view, and then select File > Properties from the Progress Developer Studio for OpenEdge menu bar.
   - Select the AppServer project in the Project Explorer view, and then right-click. From the Context menu, select Properties.

2. Select Progress OpenEdge > Modules. The Modules properties page appears.

   **Note:** By default, the AppServer Content folder that you created for the project in the New Project wizard appears in the page.

3. If you want to change the AppServer module name, select the module and click Modify. On the Modify OpenEdge AppServer Module dialog, modify the module name. You can also select which folders you want to add to or remove from the module.

4. Click OK.

5. Click OK when you finish configuring folders for the module.

Associating modules with servers

You must associate the AppServer module or set of modules with the server before you can publish code to the server. You accomplish this from the Servers view.

1. From the Servers view, select the server for which you want to associate modules.

2. Right-click, and choose Add and Remove from the Context menu. The Add and Remove dialog appears and displays a list of available projects modules and another list of configured projects modules.

3. Select a project module from the Available list, and then click Add to move the module to the Configured list.

4. If you want to move all the available project modules, click Add All. To remove an individual project module, select it in the Configured list and click Remove. To remove all project modules from the Configured list, click Remove All.

5. You can also select the option to publish changes immediately upon the startup of the server.

**See also**

Project support for AppServers on page 139
Add and Remove page on page 159
Working with facets in an AppServer project

When you create an AppServer project, additional natures, an AppServer module, and OpenEdge, AppServer, and ABL AppServer facets are added to the project.

**Note:** A nature (that allows the WTP server framework to add a module to a project) is hidden from your view, whereas a project facet is intended to be visible to and used by you. Installation of the facet creates the module. For more information about facets, modules, and natures, see the *Web Tools Platform User Guide*.

To use AppServer project facets:

1. Open the project's properties in one of the following ways:
   - Select the project in the **Project Explorer** view, and then select **File > Properties** from the Progress Developer Studio for OpenEdge menu bar.
   - Right-click the project in the **Project Explorer** view, and then select **Properties** from the **Context** menu.

2. Select **Project Facets** from the list of properties displayed in the left pane.

   **Note:** The **Project Facets** option is listed in the properties only if the facet nature has been installed in the project. Existing OpenEdge projects do not have project facets unless you migrate them to use facets.

   The **Project Facets** page opens, with the **OpenEdge** and the **AppServer** facets listed and selected.

3. If you want to remove a facet and its module definition file, clear the facet check box and click **Apply**.

   **Note:** You cannot remove the **OpenEdge** facet.

4. To reinstall the **AppServer** project facet, select it on the **Project Facets** page. The **Further configuration available** link appears near the bottom of the page.

5. Click the link. The **Modify Faceted Project** dialog box appears for the project.

6. Verify (or edit) the module name and the AppServer source folder name, and select the folders you want to publish.

7. To exit the project facet set up, click **OK**. Click **Apply**, and then click **OK** to close the **Project Facets properties** page.

**See also**

[Associating modules with servers](#) on page 152

**Publishing AppServer code to a remote server**

You can publish AppServer code to a remote server.

To publish to a remote server:

1. From the management console list frame for the remote machine (http://remote-machine-name:9090), select **Resources > remote-container-name > OpenEdge > AppServer**.

2. Select the AppServer broker you want to publish to.

3. In the detail pane, click **Configuration**.
4. From the **Configuration** page, click the **Broker** tab.

5. In the **Advanced Features** section of the page, verify that the **Publish directory property** is set to a valid location. If the property value says Not set, click **Edit** at the top of the page.

6. Type the full path to the publish directory in the **property** field.

7. Click **Save**.

8. On the local machine, create an OpenEdge Explorer connection for the remote server as follows:
   a) Select **Preferences > Progress OpenEdge > Server > OpenEdge Explorer**.
   b) Click **Add**. The **Add OpenEdge Explorer Connection** dialog opens.
   c) Type the connection name.
   d) Type the Host name or IP address.
   e) Specify the port. (The default port is 9090.)
   f) Type the user name. (The default user name is admin.)
   g) Type the password. (The default password is admin.)
   h) Provide the OpenEdge Explorer URL. (If you are using the default host name and port number, the URL is http://localhost:9090.)
   i) Click **Create Default Server** and **Finish**. If there are no default servers available in the Servers view, a default server will be created and will appear in the Servers view.

9. From the **Server Editor** view, review the Publish directory setting to verify that the correct path is shown.

---

**Publishing AppServer code to a server for testing**

Publishing code to a server so you can test it involves the following overall steps:

- **Associating modules with servers**
- **Publishing to a local or remote server**
- **Removing published content from the server**

**See also**
Publishing on page 138

---

**Using the Clean option before republishing**

You can use the **Clean** option, available in the **Servers** view, to remove any invalid resources from the server before doing a full republish.

1. Before you run the **Clean** option, decide if you want to remove all files and folders or just those from the deployed module. Then do the following:
   a) Select **Windows > Preferences > Progress OpenEdge > Server**.
   b) Select the **Remove all files and folders when cleaning server publish directory** option to remove all files. To remove only those files associated with the deployed module, be sure not to select the option.
c) Click OK.

2. To then use the Clean option:
   a) From the Servers view, select a server and right-click. The Context menu appears.
   b) Select Clean, and then click OK to confirm.

See also
Publishing on page 138
Publishing AppServer code to a server for testing on page 154

Viewing a server’s log files

The log viewer helps you to view and monitor the log files of a PAS for OpenEdge server instance from the Servers view. There are multiple log files for a server, you can choose to go to the directory where the log files are saved and manually search for errors, or view from Progress Developer Studio for OpenEdge using the log viewer.

To view the log file:

1. From the Servers view, select the PAS for OpenEdge server instance.

2. Right-click and select View log in Log viewer. The Log Viewer opens.

   The oepas.agent.log, catalina.<date>.log, oemanager.<date>.log, oepas.<date>.log, and admserv.log files are displayed by default in different tabs in the Log Viewer. Click the respective tab to view the details of a log file.

3. To display other log files in the Log Viewer, select Open Log File from the Server view menu and select the required log file.

4. To configure rules for highlighting a specific row for ERRORS or WARNINGS, select Show rule preferences from the Servers view and create rules as required.

   Note: A rule for ERROR is added by default.

Reference

AppServer-related views, pages, and wizards

This section describes the AppServer-related views, pages, and wizards including the Progress OpenEdge Server Monitor view, the New Server wizard, the Server Editor, and the Add/Trim Agents dialog.
Progress OpenEdge Server Monitor view

The Progress OpenEdge Server Monitor view provides a status summary for the AppServer broker and agents. You can access the view by:

- Opening the The AppServer in the OpenEdge Server perspective on page 137 and clicking the Progress OpenEdge Server Monitor view tab.
- From any other perspective, select Window > Show View > Other > Server > Progress OpenEdge Server Monitor.
- From the Servers view, right-click and select Progress OpenEdge Server Monitor.

Then expand the AppServer folder and select a broker.

The view provides details about the broker on a Summary tab and also includes a toolbar that you can use to perform certain AppServer tasks.

Summary tab

The following summary details related to broker and agent status appear in the Summary tab:

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broker status</td>
<td>The broker status: starting, active, stopping</td>
</tr>
<tr>
<td>Operating mode</td>
<td>The operating mode: State-reset, State-aware, Stateless, and State-free</td>
</tr>
<tr>
<td>Broker port</td>
<td>The port that the broker listens on for connections</td>
</tr>
<tr>
<td>Broker pid</td>
<td>The operating machine process ID of the broker executable</td>
</tr>
<tr>
<td>Active agents</td>
<td>The number of active agents, including all those busy, locked, and available</td>
</tr>
<tr>
<td>Busy agents</td>
<td>The number of busy AppServer agent processes</td>
</tr>
<tr>
<td>Locked agents</td>
<td>The number of locked AppServer agents that are tied to a specific client</td>
</tr>
<tr>
<td>Available agents</td>
<td>The number of available agents that are currently idle</td>
</tr>
<tr>
<td>Active clients</td>
<td>The number of active client requests (current, peak)</td>
</tr>
<tr>
<td>Client queue</td>
<td>The client queue depth of waiting clients (maximum, average)</td>
</tr>
<tr>
<td>Total Requests</td>
<td>The total number of requests serviced by this broker and all its agents</td>
</tr>
<tr>
<td>Request wait</td>
<td>The number of milliseconds clients have to wait before having their requests serviced (maximum, average)</td>
</tr>
<tr>
<td>Request duration</td>
<td>The length of time each request is taking (maximum, average)</td>
</tr>
</tbody>
</table>

Enable Agent State tab

If you have enabled the Enable Agent State tab in the Server preferences, details appear on the tab when you open it. (To open the tab, start the server in the Servers view. In the Progress OpenEdge Server Monitor view, expand the server, and select an agent from the list. The Agent State tab appears next to the Summary tab.)
If you have not enabled the tab, no details appear. Instead, a message informs you that the tab is disabled. To enable the tab, click the down arrow in the Progress OpenEdge Server Monitor toolbar, choose Preferences, and select the option.

The following information for a single AppServer agent appears in the Enable Agent State tab when you have enabled it:

<table>
<thead>
<tr>
<th>PROPATH</th>
<th>Specifies the search path that agents use to locate ABL procedures that they execute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Databases</td>
<td>Lists the name of each connected database, including its full path</td>
</tr>
<tr>
<td>Persistent procedures</td>
<td>Lists the name of each remote persistent procedure running in the Application Server process</td>
</tr>
</tbody>
</table>

**Note:** The details shown in the Enable Agent State tab do not refresh. To see refreshed information, either click Refresh server status on the view's toolbar, or set up an automatic server status refresh by setting the Server status refresh interval preference.

**Progress OpenEdge Server Monitor view toolbar**

You can use the buttons on the Progress OpenEdge Server Monitor view toolbar to accomplish the following tasks:

| ![ ] | Automatically refresh the server status at a predetermined interval. |
| ![ ] | Refresh the server status. (Available only when server is started and selected.) |
| ![ ] | Add one or more agents. (Available only when server is started and selected.) |
| ![ ] | Trim one or more agents. (Available only when server is started and selected.) |
| ![ ] | Trim all agents. (Available only when server is started and selected.) |
| ![ ] | Open a Web browser by using the currently selected server and start the OpenEdge Explorer connection the server is using. **Note:** For more information, see Accessing OpenEdge Explorer. |
| ![ ] | Access Progress OpenEdge Server preferences. |

**New Server wizard**

Use the Eclipse New Server wizard to define an AppServer connection from Progress Developer Studio for OpenEdge. You access the wizard by selecting File > New > Server from the Server perspective. The following pages appear sequentially in the wizard:

- Define a New Server page on page 158
- Define a new AppServer broker page on page 158
- Define publishing rules page on page 158
Add and Remove page on page 159

Define a New Server page
From the Eclipse Define a New Server page, choose the type of server you want to create by setting the following properties.

<table>
<thead>
<tr>
<th>Select the server type</th>
<th>Type AppServer in the filter field provided, or select Progress Software Corporation &gt; OpenEdge &gt; AppServer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server's host name</td>
<td>Type the host name. If the server is in a different domain from the local host, you must provide the full path.</td>
</tr>
<tr>
<td>Server name</td>
<td>Accept the server name, or type a different one.</td>
</tr>
<tr>
<td>Server runtime environment</td>
<td>Accept the server runtime environment name, or click Add to identify an alternate OpenEdge AVM by name and version number.</td>
</tr>
<tr>
<td>Configure runtime environments</td>
<td>Click the option if you want to edit, remove, or search for an existing runtime environment; or add a new runtime environment.</td>
</tr>
<tr>
<td>Next</td>
<td>Click to move to the Define a new AppServer broker page, the next page in the wizard.</td>
</tr>
</tbody>
</table>

Define a new AppServer broker page
From the Define a new AppServer broker page, you can identify and set the following properties for the OpenEdge Explorer connection:

<table>
<thead>
<tr>
<th>OpenEdge Explorer connection</th>
<th>Type the name of an existing OpenEdge Explorer connection, or click Configure to add a connection.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broker</td>
<td>To choose from a list of available brokers, click the dropdown arrow. To refresh the list, click Refresh.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Remember that you cannot create a new AppServer broker in Progress Developer Studio for OpenEdge; to create a broker, use OpenEdge Explorer.</td>
</tr>
<tr>
<td>Server name</td>
<td>The name of the AppServer.</td>
</tr>
<tr>
<td>Back</td>
<td>Click to move back to the Define a New Server page, the previous page in the wizard.</td>
</tr>
<tr>
<td>Next</td>
<td>Click to move to the Define publishing rules page, the next page in the wizard.</td>
</tr>
</tbody>
</table>

Define publishing rules page
From the Define Publishing Rules page, you can specify a publish location for the AppServer broker and identify the file types to be published.
Select the server publish location for a local broker. If you select the custom publish directory, provide the path to its location in the Publish directory field. Alternatively, you can click Browse to select another directory on the local machine.

Publish source code
Select to publish the source code. This option is selected by default.

Publish r-code
Select to publish the r-code. This option is selected by default.

Compile on publish
Select this option to compile code on a remote server whenever you publish.

Click to move back to the Define a new AppServer broker page, the previous page in the wizard.

Click to move to the Add and Remove page, the next page in the wizard.

Add and Remove page
From the Eclipse Add and Remove page, you can add or remove project modules that are being configured on the server.

Add
Select an available project module and click Add to move it to the list of configured project modules.

Remove
Select a configured project module and click Remove to move it to the list of available project modules.

Add All
Click to move all available project modules to the list of configured project modules.

Remove All
Click to move all configured project modules to the list of available project modules.

Click to move back to the Define publishing rules page, the previous page in the wizard.

Click to finish creating the server.

See also
Publishing on page 138
Defining an AppServer server on page 142
Defining the OpenEdge AVM Runtime on page 145
Viewing AppServer status on page 149
Publishing AppServer code to a server for testing on page 154
Accessing OpenEdge Explorer on page 147
Server Runtime Environments preferences on page 162
**Server Editor**

You use the Server Editor to view or modify the server properties that define the connection to OpenEdge Explorer and the broker.

From the Servers view in the OpenEdge Server perspective, you can access the Server Editor in either of the following ways:

- Double-click the server name.
- Select the server name, right-click, and select **Open**.

The **Server Editor** provides information in the following categories:

- **General Information** - Provides the host name and other common settings.
- **Connection** - Specifies the information for connection to OpenEdge Explorer.
- **Publishing** - Specifies when to publish.
- **Timeouts** - Specifies the time limit to complete server operations (**Start** and **Stop**).
- **Publish Location** - Specifies the server publish directory.

**General Information**

You can modify the following **General Information** properties in the **Server Editor**:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server name</td>
<td>The name of the AppServer.</td>
</tr>
<tr>
<td>Host name</td>
<td>The name of the host machine.</td>
</tr>
<tr>
<td>Runtime Environment</td>
<td>The current runtime environment. You can click the drop-down list or the <strong>Runtime Environment</strong> link to change to a different OpenEdge AVM runtime or OpenEdge version.</td>
</tr>
</tbody>
</table>

You can also open and edit the launch configuration settings for the AppServer.

**Connection**

You can modify the following OpenEdge Explorer properties in the **Server Editor**:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenEdge Explorer connection</td>
<td>The name of an existing OpenEdge Explorer connection. To add a connection, click <strong>Configure</strong>.</td>
</tr>
<tr>
<td>Broker</td>
<td>To choose from a list of available brokers, click the dropdown arrow. To refresh the list, click <strong>Refresh</strong>.</td>
</tr>
</tbody>
</table>

**Note**: Remember that you cannot create a new AppServer broker in Progress Developer Studio for OpenEdge; to create a broker, use OpenEdge Explorer.

**Publishing**

You can modify the publishing settings to choose one of the following options:

- Never publish automatically
- Automatically publish when resources change
- Automatically publish after a build event

You can also set the publishing interval in seconds.

**Timeouts**

You can specify the time limit (in seconds) for completing **Start** and **Stop** server operations.

**Publish Location**

You can set the following publish location options:

<table>
<thead>
<tr>
<th>Specify the server publish directory</th>
<th>Choose to use the server working directory or set a custom publish directory. If you choose to set the custom directory, type its location in the Publish directory field.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publish source code</td>
<td>Choose whether to publish source code; this option is selected by default.</td>
</tr>
<tr>
<td>Publish r-code</td>
<td>Choose whether to publish r-code; this option is selected by default.</td>
</tr>
<tr>
<td>Compile on publish</td>
<td>To compile code when you publish to a remote server, select this check box.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> You can select this option only if you do not select the Publish r-code option.</td>
</tr>
<tr>
<td>Compile options</td>
<td>Add any compile options, such as startup or other parameters. Specify each option with a space between its name and value, and also insert a space between each option.</td>
</tr>
</tbody>
</table>

**Add/Trim Agents dialog**

When agent requests from AppServer clients are numerous, use the Add Agents dialog to increase the number of agents running and available to service those requests. Similarly, when there is a reduction in the number of agent requests by AppServer clients, you can use the Trim Agents dialog to trim the number of agents running and available to service those requests.

You can access either dialog from the Servers view. Select an AppServer broker, and then select Add Agents or Trim Agents (or Trim all Agents) from the Context menu. Type the number of agents you want to add/trim in the Count field, and then either click OK to add/trim the agents or click Cancel to keep the number of agents as it is.

**See also**

- Adding AppServer agents on page 148
- Trimming AppServer agents on page 148

**AppServer-related preferences**

This section describes how to set AppServer-related preferences.
Server Runtime Environments preferences

You set OpenEdge runtime AVM properties in the Eclipse Server Runtime Environments preferences page. You access the page by selecting Window > Preferences > Server.

You can add, edit, or remove server runtime environments by using the following options:

<table>
<thead>
<tr>
<th>Add</th>
<th>To select a new OpenEdge AVM runtime, click Add.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit</td>
<td>To make changes to a server runtime name or OpenEdge version number, select the server runtime in the list and click Edit.</td>
</tr>
<tr>
<td>Remove</td>
<td>To remove a server runtime, select it and click Remove.</td>
</tr>
<tr>
<td>Search</td>
<td>To find an existing server runtime that is not in the list, click Search. Locate the environment, and click OK to add the server runtime to the list.</td>
</tr>
</tbody>
</table>

See also

Defining the OpenEdge AVM runtime on page 145

Progress OpenEdge Server preferences

You can set several Progress OpenEdge Server preferences related to publishing and to the Progress OpenEdge Server Monitor view.

You can access the Progress OpenEdge Server preferences page in several different ways, such as:

- From the Progress Developer Studio for OpenEdge menu bar, by choosing Window > Preferences > Progress OpenEdge > Server.
- From the Progress OpenEdge Server Monitor view, by clicking the dropdown arrow on the view toolbar and selecting Preferences.

You can set these Server preferences:

<table>
<thead>
<tr>
<th>WebSpeed Broker URL</th>
<th>Set the Web browser URL for OpenEdge Explorer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trim agents on publish</td>
<td>Select this option to trim all agents automatically from the broker whenever a publish event occurs. Whenever modules are published to the server, all the server’s agents will be trimmed immediately after the publish event succeeds. If the publish event fails for any reason, the agents will not be trimmed.</td>
</tr>
<tr>
<td>Note: This option is enabled by default.</td>
<td></td>
</tr>
<tr>
<td>Remove deleted files and folders on publish</td>
<td>Select this option to remove all files and folders from the publish location on publishing. This will remove from the server any files or folders that you have removed from the module.</td>
</tr>
<tr>
<td>Note: This option is enabled by default.</td>
<td></td>
</tr>
</tbody>
</table>
| **Remove all files and folders when cleaning server publish directory** | Select this option to remove all files and folders following the completion of clean operations.  
**Note:** This option is enabled by default. |
|---|---|
| **Publish empty folders** | Select this option if you want to publish folders even if they are empty.  
**Note:** This option is enabled by default. |
| **Run/Debug - Update properties from server before starting/launching** | Select this option to use the broker's defined properties rather than the server's launch configuration. (When a server is started, all the server's launch configuration properties are used for the broker on the remote machine. Selecting this option prevents the overwriting of the broker properties with the server properties.)  
**Note:** This option is cleared by default. |
| **Publish filters** | Select one or more items in the filter list to avoid publishing specific resources. To remove a filter, select it in the Publish Filters table and then click **Remove**.  
**Filter**  
To add a filter, click **Add** (next to the Publish Filters table) and then type the filter expression in this field. Then click **Update**. |
| **Server status refresh interval (in seconds)** | Specify the refresh delay interval for server status updates.  
**Note:** The default is 15 seconds. |
| **Enable Agent State tab** | Select this option to show the tab in the **Progress OpenEdge Server Monitor** view.  
**Note:** This option is cleared by default. |
| **Restore Defaults** | Click to restore the original set of publish filters and all other preference options on this page. |
| **Apply** | Apply the preference selections. |

**OpenEdge Explorer Connections preferences**

The OpenEdge Explorer Connections preferences page is used to maintain the available explorer connections. By default, an OpenEdge Explorer Connection profile is created when a new workspace is launched. You can manually edit the explorer connection or create a new explorer connection.

Access this page by clicking **Progress OpenEdge > Server > OpenEdge Explorer Connections** in the Preferences page.
This page includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Displays the <strong>Add Connection Profile</strong> dialog that allows you to add a new explorer connection profile.</td>
</tr>
<tr>
<td>Edit</td>
<td>Displays the <strong>Edit Connection Profile</strong> dialog that allows you to edit the selected explorer connection profile.</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes the selected explorer connection profile from your workspace.</td>
</tr>
<tr>
<td>Default</td>
<td>Allows you to select a default explorer connection profile.</td>
</tr>
<tr>
<td>Import</td>
<td>Displays the <strong>Import Workspace OpenEdge Explorer Connections</strong> dialog that allows you to import a connection profile.</td>
</tr>
<tr>
<td>Export</td>
<td>Displays the <strong>Export Workspace OpenEdge Explorer Connections</strong> dialog that allows you to export a connection profile.</td>
</tr>
</tbody>
</table>

**See also**

- [Accessing OpenEdge Explorer](#) on page 147
- [Progress OpenEdge Server preferences](#) on page 162
Overview of Progress Application Server for OpenEdge

The Progress Application Server (PAS) is a platform that provides Web server support for Progress applications. Progress applications are packaged as Web application archives (WAR files) and deployed to the Java Servlet Container of a running instance of PAS. Client access to a PAS server is through HTTP/HTTPS protocols. Clients include (but are not limited to) browser-based applications and mobile apps.

PAS for OpenEdge is a Web application server based on the Apache Tomcat Web server and servlet container. For more information on Progress Application Server, see the Progress Application Server for OpenEdge: Introducing PAS for OpenEdge guide.

For details, see the following topics:

- Concepts
- Tasks
- References
Concepts

What is Progress Application Server for OpenEdge?

The Progress Application Server (PAS) is the core Web application server, based on Apache Tomcat, that is the foundation for application servers in OpenEdge as well as other Progress products. Progress Application Server for OpenEdge (PAS for OpenEdge) is a Progress Application Server that is tailored specifically to support OpenEdge applications, including WebSpeed applications.

PAS for OpenEdge is available as two separate products:

• The Progress Production Application Server for OpenEdge — configured as a secure Web server for OpenEdge application deployment

• The Progress Development Application Server for OpenEdge — configured as a Web server for developing and testing OpenEdge applications

Note: PAS for OpenEdge and the OpenEdge AppServer are two separate and unique application servers in OpenEdge. Both support ABL application development and deployment but are entirely different in their architectures.

Application development, testing, debugging, and deployment is supported by the Progress Development Studio (PDS) for OpenEdge, which includes a project type and a development server for PAS for OpenEdge. The PDS for OpenEdge is the recommended development tool for Web applications deployed on PAS for OpenEdge. For more information, see Progress Developer Studio for OpenEdge Online Help.

For information about developing new, or migrating existing ABL applications to PAS for OpenEdge, see Progress Application Server for OpenEdge: Application Migration and Development Guide.

Administration and configuration support is provided by the OpenEdge Management (OEM) and OpenEdge Explorer tools. For more information, see OpenEdge Management: Progress Application Server for OpenEdge Configuration.

PAS for OpenEdge also includes a command line utility (TCMAN), plus APIs (JMX and REST) for creating your own administrative scripts and utilities. See Progress Application Server for OpenEdge: Administrative Guide for more information.

The PAS for OpenEdge Server perspective

The PAS for OpenEdge appears in the OpenEdge Server perspective, which includes the various views that you use to perform related activities in Progress Developer Studio for OpenEdge. The perspective is based on the OpenEdge Editor perspective but also includes the Servers view and the Progress OpenEdge Server Monitor view.

By default, the OpenEdge Server perspective includes the following views:

• Project Explorer - A hierarchical view of the projects and resources in a workspace. The view is an adaptation of the Workbench Project Explorer view with some OpenEdge-specific menu options.

• Outline - A list of the structural elements of the file that is currently open in the ABL Editor or an XML Editor. The view is the Workbench Outline view customized for working with ABL.

• DB Structure - An OpenEdge view that displays the schema of connected databases. You can drag and drop schema elements (tables, fields, and others) from this view into a file that is open in the ABL Editor.
• **Console** - A Workbench view that displays the text output from commands (runtime startup, for example) similar to the output in a command window.

• **Problems** - A log of the errors, warnings, and other information associated with a file that has been opened in the ABL editor. The view is the standard Workbench Problems view.

• **Tasks** - A To-Do list. The view is the standard Workbench Tasks view.

• **Properties** - The properties of the object currently selected in the Workbench.

• **Servers** - A list of all defined servers, including Progress Application Server. This list appears if the AdminServer is running; otherwise, the view is blank.

• **Progress OpenEdge Server Monitor** - A status view for a PAS for OpenEdge instance.

**PAS for OpenEdge launch configuration**

A launch configuration is a group of settings that govern an instance of the OpenEdge AVM. You can define the settings for one or more launch configurations for each PAS for OpenEdge instance.

For more details, see Managing launch configurations in the Running and Debugging ABL Programs help.

**Project support for PAS for OpenEdge**

Most operations that you perform with Progress Developer Studio for OpenEdge take place within the context of a project. You can create multiple projects and maintain them concurrently in your workspace.

ABL Web App and ABL AppServer projects support PAS for OpenEdge.

**ABL Web App project type for PAS for OpenEdge**

The ABL Web App project type is an OpenEdge project that lets you deploy one or more ABL services as a single web app to PAS for OpenEdge. It contains the ABL Web App, ABL AppServer, Progress Adapters, and OpenEdge facets. It contains ABL WebSpeed (for ABL Service of WebSpeed type), ABL REST (for ABL Service of REST type), or ABL Data Object (for ABL Service of Data Object type) facets depending on the ABL Service type that you created while creating the project.

After creating the project, if you define any ABL Service for that project, the corresponding facet gets added.

You can see the facets enabled for your project using the Project Facets page.

**Terminology related to PAS for OpenEdge**

The following terms describe various components that apply to PAS for OpenEdge support in Progress Developer Studio for OpenEdge:

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>An instance of a server runtime. You can define many servers for a given runtime. For PAS for OpenEdge support in Progress Developer Studio for OpenEdge, a server is used to connect to OpenEdge Explorer and provides connection information and publishing options for a particular PAS for OpenEdge.</td>
</tr>
<tr>
<td>Server configuration</td>
<td>A collection of files used to configure a running server. Typically, the configuration consists of a set of port numbers and other properties.</td>
</tr>
</tbody>
</table>
Server Runtimes (or runtimes) | The software application that executes as a server platform. For OpenEdge, this refers to the location of the OpenEdge installation that is used to communicate with OpenEdge Explorer and provide PAS for OpenEdge functionality
---|---
Modules | A collection of files used to organize content that is being developed for a server. A module usually consists of a hierarchy of files that reside in the Eclipse workspace. Modules also have a many-to-many relationship with other modules, and the file hierarchy for modules does not indicate any specific module containment. A PAS for OpenEdge project is allowed one module, and a PAS for OpenEdge module is required for publishing code to a server.
Facets | A mechanism used to add project functionality that you can turn on or off. Within OpenEdge, you can use facets to add a specific module type to a project. When you add a facet to a project, the facet is used as a marker for performing certain setup requirements or for simply marking the project as a specific project type.
ABLApplication | A logical name to groups different ABL Web applications.

For more information, see the details in overview to the *Web Tools Platform Guide*.

**Overview of WebSpeed support in PAS for OpenEdge**

PAS for OpenEdge is available as a host for WebSpeed applications. You can migrate existing WebSpeed applications to PAS for OpenEdge, or you can use Progress Developer Studio for OpenEdge to develop new WebSpeed applications that run on PAS for OpenEdge.

**Tasks**

**Working with PAS for OpenEdge**

**Defining PAS for OpenEdge**

PAS for OpenEdge is a web application server that lets you deploy all OpenEdge services on a single server. With the ABL AppServer and OE Web server, you had to deploy business logic on the AppServer broker (restbroker1), and the Data Object service on the OE Web server.

To define PAS for OpenEdge:

1. From the **Servers** view in the **OpenEdge Server** perspective, right-click and select **New > Server** from the context menu. The **Define a New Server** box appears.
2. To select PAS for OpenEdge as the server type, do one of the following:
   - Type Progress Application Server in the **Select the server type** field. This setting filters the results to display only the PAS for OpenEdge's list.
• Select Progress Software Corporation > Progress Application Server as the server type from the list in the box.

**Note:** The server’s host name, localhost, is prefilled. The Server name and the Server runtime environment details are also listed. If you want to specify a different server runtime environment, performs Steps 3 through 5. Otherwise, go to step 6.

3. Click **Add**. The Identify an OpenEdge installation box appears.

4. Enter the runtime environment name and the OpenEdge version number and then click **Finish**.

5. To set up the runtime environment, click **Configure runtime environments** and follow the steps provided in Defining the OpenEdge AVM runtime.

6. Click **Next**. The Define a new Progress Application Server box appears, in which the Server details are prefilled.

7. Do one of the following:
   • Select the name of the connection from the OpenEdge Explorer connection drop-down list.

   **Note:** Make sure that the selected OpenEdge Explorer connection has a Progress Application Server defined.

   • Click **Configure**. The Progress OpenEdge > Server > OpenEdge Explorer Connections Preferences page opens.

8. Select the Progress Application Server for OpenEdge and the default ABL Application.

9. Click **Next**. The Add and Remove box appears. It displays the available project modules (OE project, REST, or Data Object service module).

10. To configure an available project module, select the module in the **Available** list and click **Add** or double-click the module in the Available list. To configure all the available project modules, click **Add All**.

   **Note:** You can remove a project module from the Configured list by selecting the module and by clicking **Remove** or double-clicking the module in the Configured list. You can also select more than one project module and click **Remove All**.

11. Click **Finish**. ABL Service modules and the business logic is published to the PAS for OpenEdge instance.

**See also**
- What is Progress Application Server for OpenEdge? on page 166
- Terminology related to PAS for OpenEdge on page 167

**Defining a Progress Application Server’s runtime environment**

The Server Runtime Environments’ Preferences page lets you define PAS for OpenEdge’s runtime environment.

To define PAS for OpenEdge’s runtime environment:
1. Click **Window > Preferences**. The **Preferences** page appears.

2. In the left pane, expand **Server > Runtime Environments**. The **Server Runtime Environments** page appears in the right pane.

3. Click **Add**. The **New Server Runtime Environment** box appears.

4. From the server runtime environment list, click **Progress Software Corporation > Progress Application Server**. The **Identify an OpenEdge installation** box appears.

5. Enter the runtime environment name and the OpenEdge version number and then click **Finish**.

**Renaming a PAS for OpenEdge instance**

You can rename a PAS for OpenEdge instance.

To rename a server:

1. From the **Servers** view in the **OpenEdge Server** perspective, right-click the required Progress Application Server instance and then click **Rename** from the context menu.

2. Type the new name of the Progress Application Server, and press **Enter**.

**Publishing modules on PAS for OpenEdge**

You can use the **Servers** view in the OpenEdge Server perspective to create a PAS for OpenEdge instance and publish the ABL Services to PAS for OpenEdge. The OpenEdge projects with the selected ABL AppServer facets, ABL services of the REST, Data Object (WebHandler), Data Object (REST RPC), or WebSpeed types are deployed on the ABL application publish directory of the selected PAS for OpenEdge.

You must associate ABL service module with a PAS for OpenEdge instance, and then publish the service. To publish the business logic present under AppServer, you must associate ABL AppServer content module with a PAS for OpenEdge instance.

**Note:** If the **Automatically publish when resources change** (default) option is selected, the server automatically tries to publish the changes after the specified publishing interval (which, by default, is set to 15 seconds). You can change this value using the **Publishing interval** control in the Server Editor, which is displayed by double-clicking the server. For more information on changing the publishing settings, see PAS Server Editor.

The ABL services are deployed using PASOEContent to build .WAR files. A single .WAR file is created for all Data Object and REST services in a project and gets deployed on basis of the project, for example if a project has ten services and a single .WAR file with the project name containing all the services gets deployed. If the project .WAR file exists, the services are published incrementally or else the entire .WAR file is published.

**To publish an ABL Service:**

1. In the **Servers** view, right-click a PAS for OpenEdge server instance that you want the ABL service to be deployed to, and select **Add and Remove**. The **Add and Remove** page opens.

2. Select an ABL AppServer content module and ABL services of WebSpeed, REST, or Data Object (WebHandler), Data Object (REST RPC) type from the **Available** section, and click **Add** to move it to the **Configured** section.

3. Click **Finish**. The web application is published as **WebAppName.war** to the **webapps** folder of the selected PAS for OpenEdge server instance. It is published to the default ROOT web application depending on what you have set in the project properties page.
Note:
If you deployed a WebSpeed service, the openedge.properties file in the `pas Instance/conf` path is updated with the WebSpeed services related handlers and URI information in the corresponding Web App section as in the example below:

```
[oepas1.WebApp.WEB]
handler1 = HandlerClass: /uri
handler2 = HandlerClass2: /uri2
```

If you deployed a WebSpeed based Data Object service, the following artifacts related to the service are deployed to the PAS instance:

- Data Service Catalog called `service.json` in WEB-INFS/OpenEdge folder.
- Generated mapping file called `service.gen` in the static folder.

For more information on these files, see Generated files.

The openedge.properties file in the `pas Instance/conf` path is updated with the following entry under the webApps' WEB section:

```
handler1=OpenEdge.Web.DataObject.DataObjectHandler: /pdo/
```

`DataObjectHandler` is the new `OpenEdge.Web.DataObject.DataObjectHandler` available in the existing `OpenEdge.Net.pl` library in both source and r-code forms. This handler uses the above generated `service.gen` file, and translates a HTTP request into a call to a business entity.

If you undeploy a WebSpeed based Data Object service, the `service.json` and `service.gen` files are removed from the server.

If you want to publish already associated module/service(s), do one of the following to publish:

- Select Publish from the context menu
- Select the Publish to the server icon on the Servers view toolbar

If all assigned or added modules on server are published successfully, then the Servers view displays the status of the server as Started, Synchronized, and the status of the server modules as Synchronized, Published.

Note:
If ABL AppServer modules are published to PAS for OpenEdge, the artifacts get deployed on the ABL application publish directory of the selected PAS for OpenEdge.

When REST services are published to PAS for OpenEdge, `service.paar` files are deployed to the server. When REST based Data Object Services are published, `service.paar` and `service.json` files are deployed to the server.

You can change the publish directory by changing the value in the Publish directory field in the Server launch configuration wizard startup tab, the default location is `pas instance/openedge`.
Deleting a PAS for OpenEdge instance
You can use the Delete option from the Servers view to remove a PAS for OpenEdge instance from the Servers view.

To delete a PAS for OpenEdge instance:
1. From the Servers view in the OpenEdge perspective, right-click the required PAS for OpenEdge instance and then click Delete from the context menu.
2. Click Yes to confirm.

   The PAS for OpenEdge and all its services and resources are deleted from the Servers view.

Viewing PAS for OpenEdge log files
You can view the log information of a currently running PAS for OpenEdge in the Console view. The log information includes server-related errors, warning messages, and informative messages. Progress Developer Studio for OpenEdge stops logging this information when the server is stopped or removed from the Servers view.

   The Console view is available when you start or restart a server from within Progress Developer Studio for OpenEdge.

You can open the Console view to display the PAS for OpenEdge log file in one of the following ways:

   • From the Servers view, select the PAS for OpenEdge whose log information you want to view and then select Show In > Console from the context (right-click) menu.
   • From the Progress Developer Studio for OpenEdge menu bar, select Window > Show View > Other > General > Console, click Display Selected Console from the Console view toolbar, and then select the server log from the drop-down list.

Working with PAS for OpenEdge launch configuration
You can run a PAS for OpenEdge instance from within the Progress Developer Studio for OpenEdge environment using the PAS for OpenEdge launch configuration. Each time you do so, Progress Developer Studio for OpenEdge runs a launch configuration, which you can customize to suit the needs of your application and its target implementation.

Starting a PAS for OpenEdge instance at runtime loads the properties defined for the server instance, the ABL applications, and the application properties (the ABL application properties are defined in <openedgeWork>/<instancename>/conf/openedge.properties). The session manager and agents are configured with the ABL Application. You can use the PAS for OpenEdge launch configuration to edit the session and agent properties.

Accessing the Progress Application Server for OpenEdge launch configuration
To access the Progress Application Server for OpenEdge launch configuration wizard, do one of the following:

   • Select the PAS for OpenEdge instance that you want to run and then select Open from the context (right-click) menu.
      Then, in the server editor, select the Launch configuration link.

   • Click the drop-down arrow next to Run to open a menu from which you can start an existing configuration, create, and run a default configuration.

   The Progress Application Server for OpenEdge launch configuration appears.
Working with PAS for OpenEdge for WebApp

Creating an ABL Web App project

To deploy one or more ABL services as a single Web App to PAS for OpenEdge, create an ABL Web App project.

**Note:** You can also use this project type to build an AppServer Internet Adapter (AIA) based application that uses the APSV transport and can be deployed to PAS for OpenEdge.

To create an ABL Web App project:

1. Select File > New > OpenEdge Project from the Progress Developer Studio for OpenEdge main menu bar or the Project Explorer view context (right-click) menu. The New OpenEdge Project wizard opens and displays the Create an OpenEdge Project page on page 105.

2. Type a name in the Project name field.
   
The root folder for the new project is created in your current Eclipse workspace, unless you specify a different location.

3. To choose a different location for the project, click and clear the Use default check box. Then either browse to, or enter the path to the desired location.

4. Select Server from the left section.

5. Do one of the following:
   
   • To create an ABL Web App project with WebSpeed (WebHandler) service, select the PAS for OpenEdge server and WEB transport.
   
   • To create ABL Web App project with REST (Mapped RPC) service, select the PAS for OpenEdge server and REST transport.
   
   • To create ABL Web App project with DataObject service and WebSpeed WebHandler service provider, select the PAS for OpenEdge server, WEB transport, and Create Data Object Service check box.
   
   • To create ABL Web App project with DataObject service and REST RPC service provider, select the PAS for OpenEdge server, REST transport, and Create Data Object Service check box.

6. Click Next. The Provide ABL Web App deploy details dialog box opens.

7. Select one of the following:
   
   • Deploy as a default (ROOT) option button if you want the WAR file to be placed in the ROOT Web App folder when you publish the project. The project is published incrementally.
   
   • Deploy as WebApp option button if you want the WAR file to be published with the Web App name that you provide.

8. Select the Business Logic check box to generate the AppServer folder in the Project Explorer.

9. Enter a Module name for the for the ABL AppServer module.

10. Enter a name of the ABL AppServer module content directory in the ABL source folder.

11. From the Supported servers list, select the servers that will be the destination for your published Web App. The primary function of these servers is to support the publishing of your code for running, testing, and deploying your Web App to PAS for OpenEdge.
Note: If you publish the AppServer module to a PAS for OpenEdge instance, all the content in the AppServer folder gets published to the <pasInstance>/openedge folder and for traditional AppServers, all the content in the AppServer folder gets published to the <WRK> directory.

12. Click Next. The Create an ABL Service page opens.
13. Follow the steps to create the ABL service.
15. Specify the OpenEdge AVM options and the project layout options (or leave the default settings).
16. Click Next. The Define PROPATH dialog box appears.
17. Retain the default settings or add the PROPATH entries to be used by the project and click Next. The Select database connections dialog box appears.
18. Select the database connections to be used by the project and click Finish.

The project is deployed to the selected PAS for OpenEdge instances with the Web App name you provided, or to the default ROOT folder incrementally based on your selection in Step 5. If you do not select any server in Step 8, the Web App is not deployed.

Creating an ABL Service

You can create ABL Services using the New ABL Service wizard and deploy them to PAS.

1. Select File > New > ABL Service from the Progress Developer Studio for OpenEdge main menu bar. The New ABL Service wizard opens.

   Note: You can also right-click an ABL Web App project, and then select New > ABL Service from the context menu. You can also access the New ABL Service wizard while creating the ABL Web App project.

2. By default, the Project field displays the currently open project to which you can associate the ABL service. Click Browse(...) if you want to select a project other than the current one.
3. Enter a Service name to display the name of the service.
4. To create an ABL Service, do one of the following:
   • To create an ABL Service of WebSpeed (WebHandler) type:
     1. Select WEB transport type.
     2. Enter a name in Create new to create a new Web handler or click Select existing and then click Browse to add an existing Web handler.
     3. Click Add to enter a Resource URI.

   Note:
The Resource URI must start with "/".
You can edit or remove a resource URI using Edit and Remove. Use the Up and Down keys to sort the resource URIs.
When you finish creating the ABL service, the WebSpeed service is generated and appears under the Defined Services node.
• To create an ABL Service of REST (Mapped RPC) type:
  1. Select the REST transport type.
  2. Enter a relative path for all the resources to be defined in Service relative URI. For example, /Customer.

  **Note:** The relative URI must start with "/".

  3. (Optional) Enter Service description for the REST service.

• To create an ABL Service of Data Object type with the WebSpeed WebHandler service provider:
  1. Select the WEB transport type.
  2. Select the Create Data Object Service check box.
  3. Enter a name in Create new to create a new Web handler or click Select existing and then click Browse to add an existing Web handler.
  4. Click Add to enter a Resource URI.

  **Note:**
  The Resource URI must start with "/".
  You can edit or remove a resource URI using Edit and Remove. Use the Up and Down keys to sort the resource URIs.
  When you finish creating the ABL service, Data Object service with WebSpeed WebHandler service provider and service.json and service.gen files are created and appears under the Defined Services node.

• To create an ABL Service of Data Object type with the REST RPC service provider:
  1. Select the REST transport type.
  2. Create the Create Data Object Service check box.
  3. Enter a relative path for all the resources to be defined in Service relative URI. For example, /Customer.

  **Note:**
  The relative URI must start with "/".
  When you finish creating the ABL service, Data Object service with REST service provider and a service.json file is created and appears under the Defined Services node.

  4. (Optional) Enter Service description for the REST service.

  5. Click Finish.

**Editing an ABL Service**

The Edit ABL Service page allows you to edit an ABL service of WebSpeed type, REST type, or Data Objects type as required.
To edit an ABL service:

1. From the Project Explorer view, expand the Defined Services node under your ABL Web App project.
2. Right-click an ABL WebSpeed service, ABL REST service or a Data Object Service, and then select Edit from the context menu. The Edit ABL Service dialog box appears.
3. Do one of the following:
   a) To edit an ABL service of WebSpeed type, click Add to enter a Resource URI.

   **Note:**
   You cannot modify the Service type, Service name or WebHandler fields.
   You can edit or remove a resource URI using Edit and Remove. Use the Up and Down keys to sort the resource URIs.

   b) To edit an ABL service of Data Object type with a REST service provider, enter a Service relative URI and Service description as required. Click Next and select the resources.

   c) To edit an ABL service of Data Object type with a WebSpeed WebHandler service provider, enter a Service description as required. Click Next and select the resources.

   d) To edit an ABL service of REST type, see in REST help.

4. Click Finish.

Creating a WebHandler Class

The web-disp.p control program available for WebSpeed applications for traditional servers does not exist in PAS for OpenEdge. Web object execution is controlled by a built-in handler object. This default handler class can be modified to implement any web-disp.p customizations that you want to replicate. However, you cannot migrate web-disp.p to a PAS for OpenEdge instance.

To modify the default handler class, create a new WebHandler class.

To create a new WebHandler class:

1. Select File > New > WebHandler from the Progress Developer Studio menu bar. The New WebHandler wizard opens.

   **Note:** You can alternatively right-click the project and select New > WebHandler.

2. Specify an open project that will contain the class code and other project code in the Package root field. Click Browse if you want to select a project other than the current one (the default value).

3. Specify a package name in the Package field that corresponds to a sub-folder of the package root, and will contain the class file. Click Browse if you want to select a project other than the current one (the default value). A period (.) must separate each subfolder from its parent folder in the path name.

4. Specify the name of the Web handler class in the WebHandler name field.

5. Select the Final check box if you do not want to allow inheritance from this Web handler class. A FINAL option is included in the generated ABL code. A Final class cannot be Abstract.

6. Select the Abstract check box to specify that the Web handler class is abstract and cannot be instantiated. An abstract class is designed to serve as a super class from which other classes inherit and implement members. Therefore, an Abstract class cannot be Final.
7. Select the **Widget pool** check box to specify that a USE-WIDGET-POOL option is included in the generated ABL code.

8. Select the **Serializable** check box to mark the Web handler class as serializable by including the **SERIALIZABLE** modifier in the generated ABL code.

**Note:** If the class inherits from a super class that is not serializable, then it cannot be marked as serializable. The WebHandler class should always inherit ‘OpenEdge.Web.WebHandler’ directly or through other classes.

9. Specify another class in the current project as a super class (or a different project that is available in the propath) from which the new class inherits state and behavior in the **Inherits** field. Click **Browse** or enter the class path by typing. A period (.) must separate each node from its parent folder in the path name.

**Note:** By default, the super class name is populated as OpenEdge.Web.WebHandler. The super class cannot be generic.

10. Specify one or more interfaces in the current project or a different project that is available in the propath that the class implements using the **Implements** section. Click **Add** and select the required interfaces in the **Interface Selection** dialog. Click **Remove** to remove an interface from the list after adding it.

11. Select the methods in the **Method stubs** section that you want to generate in the Web handler class.

**Note:** The GET method is selected by default.

12. Select one of the following error-handling statements to add it to the generated ABL code:

   - **Block level**: Includes the **BLOCK-LEVEL ON ERROR UNDO, THROW** statement in the generated ABL code.
   - **Routine level**: Includes the **ROUTINE-LEVEL ON ERROR UNDO, THROW** statement in the generated ABL code.

13. Select **Throw a Not Implemented exception** option button to specify that "METHOD NOT IMPLEMENTED" error code is added to stubs for members implemented via interface and stubs for inherited abstract members.

14. Select **Return a default value** option button to specify that the default return values are added to stubs for members implemented through interface and stubs for inherited abstract members.

15. Click **Finish**.

### Exporting an ABL Service

The **Export ABL Web Application** wizard allows you to package an ABL service as a .WAR file to deploy as a complete web application or to export services incrementally by creating a .ZIP file that contains .PAAR files for each selected ABL REST and ABL Data Object (REST RPC) and .gen and pdo.handlers file for ABL Data Object (WebHandler). If you selected WebSpeed services, .properties or .handlers files are created. You can use the .WAR file or the .ZIP file to deploy the packaged Web application to a remote server.

To export as ABL Web Application:

1. In the **Project Explorer** view, select the ABL Web App project that contains the ABL services that you want to export as a Web application.

2. Right-click the project and click **Export** from the context menu. The **Export ABL Web Application** dialog box opens.
3. The **Project** drop-down list displays the project you have selected in Step 1. You can select a different project from the drop-down list.

**Note:** The **Project** drop-down displays only the ABL Web App projects in your current workspace.

4. In the **Destination** drop-down list, specify a location or click drop-down displays only the ABL Web App projects in your current workspace. **Browse** to select a different location for the Web application.

5. Select one of the following options:

- **Export ABL services for Progress Application Server** to export the services by creating a single .WAR file for all the selected ABL services in the project.

- **Export services for incremental deployment** to export services by creating a .ZIP file that contains .PAAR file for each selected ABL REST, .PAAR and .json files for each selected REST based Data Object Service, .handlers files for selected ABL WebSpeed services, and pdo .handlers and .gen files of WebSpeed based Data Object Services.

6. Select the **Service Name** check box to select all the services in the project or click the individual ABL services that you want to export.

7. Click **Finish**.

When you export an ABL Service of any type fully, a .WAR file with PASOE content folder of ABL WebApp project is exported with some other files depending on the ABL Service type as follows:

- When you export ABL Service of REST type, .PAAR files for each selected service is generated in the WEB-INF/adapters/rest/servicename location.

- When you export ABL Service of WebSpeed type, merge.properties files is generated in the WEB-INF/tlr location. This file contains multiple entries for each handler of the selected services:

```java
[${oepas-app}.${oepass-webapp}.WEB]
adapterEnabled=1
handler1=sampleHandlerClass:/uri1
srvrDbug=0
```

- When you export REST based ABL Service of Data Object type, .PAAR files for each selected service is generated in the WEB-INF/adapters/rest/servicename location and servicename.json

- When you export WebSpeed based ABL Service of Data Object type, merge.properties file is generated the file in the static folder WEB-INF/tlr file in the location with a single entry as follows:

```java
handler1=OpenEdge.Web.DataObject.DataObjectHandler:/pdo/
```

*servicename.json in the static folder and servicename.gen file in WEB-INF/openedge is generated*
When you export an ABL Service of any type incrementally, a .ZIP file containing the static folder of ABL WebApp project and the openedge folder in the Web-INF folder is exported with some other files depending on the ABL Service type as follows:

- When you export an ABL Service of REST type, .PAAR files for each selected service is generated in the WEB-INF/adapters/rest/servicename location.

- When you export an ABL Service of WebSpeed type, servicename.handlers file for each service is created in in WEB-INF/openedge location. Here is an example of .handlers file:

  ```json
  {
    "version": "1.0",
    "servicename": "sampleWebSpeedservice",
    "handlers": [{
      "className": "SamplehandlerClass",
      "URIs": [
        "\uri1",
        "\uri2"
      ]
    }]
  }
  ```

- When you export a REST based ABL Service of Data Object type, .PAAR files for each selected service is generated in the WEB-INF/adapters/rest/servicename location and servicename.json file in the static folder.

- When you export a WebSpeed based ABL Service of Data Object type, servicename.json file in Static folder and servicename.gen file in the WEB-INF/openedge location are generated. A single pdo.handlers file for all the selected services is generated in the WEB-INF/tlr location as follows:

  ```json
  {
    "version": "1.0",
    "serviceNam"e: "pdo",
    "handlers": [{
      "className": "OpenEdge.Web.DataObject.DataObjectHandler",
      "URIs": ["\pdo\"]
    }
  }
  ```

**Migrating a REST Project to ABL Web App Project**

You can migrate an existing traditional REST project and all its services to an ABL Web App project to deploy the project to PAS for OpenEdge instance.

1. Remove all the services from the REST manager and REST Broker that are related to the project that you are migrating.

2. Select the project and click **Properties > Project Facets**. The Project Facets page opens.

3. Clear the REST and ABL AppServer facets check boxes.

4. Click **Apply**.

5. Select ABL AppServer, ABL Web App, and ABL REST Service facets check boxes.
6. Click **Apply**.
7. Click **OK**. The project is now an ABL Web App project

**Migrating a Data Object Project to ABL Web App Project**

You can migrate an existing traditional Data Object project and all its services to an ABL Web App project to deploy the project to PAS for OpenEdge instance.

1. Remove all the services from the REST manager and REST Broker that are related to the project that you are migrating.
2. Select the project and click **Properties > Project Facets**. The Project Facets page opens.
3. Clear the Data Object, REST and ABL AppServer facets check boxes.
4. Click **Apply**.
6. Click **Apply**.
7. Click **OK**. The project is now an ABL Web App project

**Migrating an ABL Web App Project**

If you have imported an ABL Web App project from OpenEdge 11.6 to the current release in the Progress Developer Studio for OpenEdge, you can also migrate the Spring Security files.

**Note:** You will not be allowed to export or deploy the 11.6 ABL Web App project until you migrate.

1. Right-click the ABL Web App project and select **Progress OpenEdge > Migrate Spring Security Files**. The Migrate Spring Security Files dialog box also opens.

   **Note:** The dialog box also opens when you open an old workspace or import an old project to 11.7 workspace that contains ABL Web App projects.

2. Click **OK**.

   **Note:** If you click **Cancel**, you must later follow the previous step to migrate.

   The Spring Security files in the current WEB-INF folder in the project directory are moved to the newly created backup folder. The migrated Spring Security files are placed in the WEB-INF folder.

**Working with Web UI Project Type**

Web UI project is a new OpenEdge project type that allows you to develop web front-end applications and deploy it on the PAS OE server. It supports deploying the build output of Kendo UI Builder project to the selected PAS instance.

A new facet, Web UI, is added to the project properties when you create a Web UI project.
Creating a Web UI project

You can use the Web UI project to develop web front-end applications deployed on PAS for OpenEdge. You can also use the Web UI project to deploy the applications created in Kendo UI Builder.

To create a Web UI project:

1. Select File > New > OpenEdge Project from the Progress Developer Studio for OpenEdge main menu bar or the Project Explorer view context (right-click) menu. The New OpenEdge Project wizard opens and displays the Create an OpenEdge Project page on page 105.
2. Type a name in the Project name field.
   The root folder for the new project is created in your current Eclipse workspace, unless you specify a different location.
3. To choose a different location for the project, click and clear the Use default check box. Then either browse to, or enter the path to the desired location.

   **Note:** If you want to use a Kendo UI Builder project with the Web UI project, ensure that the name of the project is same as the Kendo UI Builder app, and you select the same location for the Web UI project where the Kendo UI Builder project resides.

4. Select Client from the left section.
5. Select the Web Application option button and select the Web UI option button.
6. Click Next.
7. Type a name in the WebApp name field.
8. From the Supported servers list, select the servers that will be the destination for your published application. The primary function of these servers is to support the publishing of your code for running, testing, and deploying your Web App to PAS for OpenEdge.
9. Click Finish. The project is deployed to the selected PAS for OpenEdge instances with the name you provided. If you have associated this project with Kendo UI Builder, the debug folder is added to the build output folder in the Project Explorer to contain all Kendo UI Builder content.

Exporting a Web UI project

You can export a Web UI project as a WAR file to create an archive of a project that can be distributed to other systems or to other users.

To export the Web UI project:

1. In the Project Explorer view, select the required Web UI project that you want to export.
2. Right-click the project and click Export > Web UI Application from the context menu. The Export Web UI Application dialog box opens.
3. The Project drop-down list displays the project you have selected in Step 1. You can select a different project from the drop-down list.

   **Note:** The Project drop-down displays only the Web UI projects in your current workspace.

4. In the Destination drop-down list, specify a location or click Browse to select a different location for the export.
5. Select one of the following options:
   - **Export Release Build** to export the contents of the Release folder that contains the production build used for publishing in the Kendo UI Builder structure in the Project Explorer.
   - **Export Debug Build** to export the contents of the Debug folder that contains the development build used for debugging the Kendo UI Builder structure in the Project Explorer.

6. Click **Finish**. A .WAR file for the selected contents in the project is exported as a .WAR file.

**Migrating an existing project to a Web UI project**

You can migrate an existing project, and all its services, to a Web UI project to deploy the project to PAS. To migrate:

1. Remove all the services from the existing projects that are related to the project that you are migrating.
2. Select the project and click **Properties > Project Facets**. The **Project Facets** page opens.
3. Clear the facets related to the current project.

   **Note:** You cannot migrate any project that has ABL AppServer 11.6, AppServer 11, Dynamics, or WebSpeed facets.

4. Click **Apply**.
5. Select **Web UI facet** check box.
6. Click **Apply**.
7. Click **OK**. The project is now a Web UI project.

**Publishing a Web UI project**

The Web UI project is published to the PAS instance that you selected while creating the project. You can also publish it to another PAS instance using the **Add and Remove** dialog box in the **Servers** view.

**Note:** For more information on changing the publishing settings, see **PAS Server Editor**.

1. In the **Servers** view, right-click a PAS instance that you want the project to be deployed to, and select **Add and Remove**. The **Add and Remove** page opens.
2. Select the Web UI project and click **Add** to move it to the **Configured** section.
3. Click **Finish**. The project and its content is published to the selected PAS instance.

If you want to publish already associated project, do one of the following to publish:

- Select **Publish** from the context menu
- Select the **Publish to the server** icon on the Servers view toolbar

If all assigned or added modules on server are published successfully, then the **Servers** view displays the status of the server as **Started**, **Synchronized** and the status of the server modules as **Synchronized**, **Published**.

**Note:** A single .WAR file is created for each Web UI project. If the project is already published, then any changes to the Web UI project are published incrementally.
PAS Server Editor

You can use the **PAS Server Editor** to view or modify PAS instance properties for an ABL Service or a Web UI project.

From the **Servers** view in the OpenEdge Server perspective, you can access the **PAS Server Editor** in the following ways:

- Double-click a PAS Server instance
- Select a PAS Server instance, right-click, and then select **Open**

The **PAS Server Editor** provides information in the following categories:

- **General Information** — Provides the host name and other common settings
- **Connection** — Specifies the information for connection to the OpenEdge Explorer
- **Publishing** — Specifies the publish settings
- **Timeouts** — Specifies the time limit to complete server operations (Start and Stop)

**General Information**

You can modify the following General Information properties in the **PAS Server Editor**:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server name</td>
<td>The name of the PAS Server instance.</td>
</tr>
<tr>
<td>Host name</td>
<td>The name of the host machine.</td>
</tr>
<tr>
<td>Runtime Environment</td>
<td>The current runtime environment. You can click the drop-down list or the Runtime Environment link to change to a different OpenEdge AVM runtime or OpenEdge version.</td>
</tr>
</tbody>
</table>

**Connection**

You can modify the following OpenEdge Explorer properties in the **PAS Server Editor**:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenEdge Explorer connection</td>
<td>The name of an existing OpenEdge Explorer connection. To add a connection, click <strong>Configure</strong>.</td>
</tr>
<tr>
<td>Progress Application Server for OpenEdge</td>
<td>To choose from a list of available PAS servers, click the drop-down arrow. To refresh the list, click <strong>Refresh</strong>.</td>
</tr>
</tbody>
</table>

**Note:** Remember that you cannot create a new PAS server in Progress Developer Studio for OpenEdge; to create a PAS server, use OpenEdge Management.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABL Application</td>
<td>The name of the ABL Application that is to be deployed to the selected PAS server instance.</td>
</tr>
</tbody>
</table>
Publishing
You can modify the publishing settings to choose one of the following options:

• Never publish automatically
• Automatically publish when resources change
• Automatically publish after a build event

You can also set the publishing interval in seconds.

Timeouts
You can specify the time limit (in seconds) for completing Start and Stop server operations.

Note: You can start and stop the PAS Server from the Servers view using the Start and Stop options on the PAS Server's context menu.

Publish source code
You can select this option to publish the source code from the folders that are in the project propath.

Publish r-code
You can select this option to publish the r-code code of the source files from the folders that are in the ABL Web App project propath.

Launching the OpenEdge web app
You can launch the app from the Server view.

To launch an app:

1. In the Servers view, expand the PAS for OpenEdge server instance.
2. Right-click the WEB UI module and click Launch App. The following URL opens and launches the app.

   http://host:port/WebApp

Undeploying a Web UI project
The Remove context option in the Add and Remove dialog box allows you to undeploy a Web UI module from a PAS instance.

To undeploy a Web UI module:

1. From the Servers view, select a PAS server instance from which you want to undeploy the Web UI module.
2. Select Add and Remove on the PAS server context menu. The Add and Remove dialog appears and displays a list of deployed Web UI module.
3. Select the Web UI module from the Available list, and then click Remove. This moves the selected Web UI module to the Configured list.

   Note: To remove all the deployed Web UI modules, click Remove All.
4. Select the **If server is started, publish changes immediately** check box to publish changes immediately upon the startup of PAS.

5. Click **Finish** to save your changes.

**Deleting a published Web UI project**

You can use the **Delete** option to remove a Web UI project.

**Note:** Make sure you want to delete the Web UI project as it is also associated with your Kendo UI Builder app.

To delete a Web UI project:

1. From the Project Explorer view, right-click the Web UI project.
2. Select **Delete** from the context menu. A confirmation dialog prompts you to confirm the deletion.
3. Click **Yes** to confirm the deletion.

The selected Web UI project and its artifacts are deleted from the workspace. The `WebAppName.war` file and the `WebAppName` folder are also deleted from the PAS instance `webapps` folder in the OpenEdge installation folder. The current workspace is automatically refreshed.

**References**

**PAS for OpenEdge-related preferences**

**Server Runtime Environments preferences**

You set OpenEdge runtime AVM properties in the **Eclipse Server Runtime Environments preferences** page. You access the page by selecting **Window > Preferences > Server**.

You can add, edit, or remove server runtime environments by using the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>To select a new OpenEdge AVM runtime, click <strong>Add</strong>.</td>
</tr>
<tr>
<td><strong>Edit</strong></td>
<td>To make changes to a server runtime name or OpenEdge version number, select the server runtime in the list and click <strong>Edit</strong>.</td>
</tr>
<tr>
<td><strong>Remove</strong></td>
<td>To remove a server runtime, select it and click <strong>Remove</strong>.</td>
</tr>
<tr>
<td><strong>Search</strong></td>
<td>To find an existing server runtime that is not in the list, click <strong>Search</strong>. Locate the environment, and click <strong>OK</strong> to add the server runtime to the list.</td>
</tr>
</tbody>
</table>

**See also**

[Defining the OpenEdge AVM runtime](#) on page 145
Progress OpenEdge Server preferences

You can set several Progress OpenEdge Server preferences related to publishing and to the Progress OpenEdge Server Monitor view.

You can access the Progress OpenEdge Server preferences page in several different ways, such as:

- From the Progress Developer Studio for OpenEdge menu bar, by choosing Window > Preferences > Progress OpenEdge > Server.
- From the Progress OpenEdge Server Monitor view, by clicking the dropdown arrow on the view toolbar and selecting Preferences.

You can set these Server preferences:

<table>
<thead>
<tr>
<th>WebSpeed Broker URL</th>
<th>Set the Web browser URL for OpenEdge Explorer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trim agents on publish</td>
<td>Select this option to trim all agents automatically from the broker whenever a publish event occurs. Whenever modules are published to the server, all the server's agents will be trimmed immediately after the publish event succeeds. If the publish event fails for any reason, the agents will not be trimmed.</td>
</tr>
<tr>
<td>Note: This option is enabled by default.</td>
<td></td>
</tr>
<tr>
<td>Remove deleted files and folders on publish</td>
<td>Select this option to remove all files and folders from the publish location on publishing. This will remove from the server any files or folders that you have removed from the module.</td>
</tr>
<tr>
<td>Note: This option is enabled by default.</td>
<td></td>
</tr>
<tr>
<td>Remove all files and folders when cleaning server publish directory</td>
<td>Select this option to remove all files and folders following the completion of clean operations.</td>
</tr>
<tr>
<td>Note: This option is enabled by default.</td>
<td></td>
</tr>
<tr>
<td>Publish empty folders</td>
<td>Select this option if you want to publish folders even if they are empty.</td>
</tr>
<tr>
<td>Note: This option is enabled by default.</td>
<td></td>
</tr>
<tr>
<td>Run/Debug - Update properties from server before starting/launching</td>
<td>Select this option to use the broker's defined properties rather than the server's launch configuration. (When a server is started, all the server's launch configuration properties are used for the broker on the remote machine. Selecting this option prevents the overwriting of the broker properties with the server properties.)</td>
</tr>
<tr>
<td>Note: This option is cleared by default.</td>
<td></td>
</tr>
</tbody>
</table>
Publish filters

Select one or more items in the filter list to avoid publishing specific resources. To remove a filter, select it in the Publish Filters table and then click Remove.

Filter

To add a filter, click Add (next to the Publish Filters table) and then type the filter expression in this field. Then click Update.

Server status refresh interval (in seconds)

Specify the refresh delay interval for server status updates.

Note: The default is 15 seconds.

Enable Agent State tab

Select this option to show the tab in the Progress OpenEdge Server Monitor view.

Note: This option is cleared by default.

Restore Defaults

Click to restore the original set of publish filters and all other preference options on this page.

Apply

Apply the preference selections.

OpenEdge Explorer Connections preferences

The OpenEdge Explorer Connections preferences page is used to maintain the available explorer connections. By default, an OpenEdge Explorer Connection profile is created when a new workspace is launched. You can manually edit the explorer connection or create a new explorer connection.

Access this page by clicking Progress OpenEdge > Server > OpenEdge Explorer Connections in the Preferences page.

This page includes the following options:

Add

Displays the Add Connection Profile dialog that allows you to add a new explorer connection profile.

Edit

Displays the Edit Connection Profile dialog that allows you to edit the selected explorer connection profile.

Remove

Removes the selected explorer connection profile from your workspace.

Default

Allows you to select a default explorer connection profile.
Import | Displays the **Import Workspace OpenEdge Explorer Connections** dialog that allows you to import a connection profile.

Export | Displays the **Export Workspace OpenEdge Explorer Connections** dialog that allows you to export a connection profile.

---

**See also**

[Accessing OpenEdge Explorer](#) on page 147

[Progress OpenEdge Server preferences](#) on page 162

---

**ABL Service Project Property page**

The ABL Service project property page helps you to set up the deploy options after you create a new ABL service.

This page appears when you right-click a project and select **Properties**.

**Note:** The options that you select, will apply to all the services that you created for this project.

The following options are available:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy as a default (ROOT)</td>
<td>Select if you want the services to be placed in the ROOT Web App folder when you publish the project. The services are published incrementally.</td>
</tr>
<tr>
<td>Deploy as WebApp</td>
<td>Select if you want the .WAR file to be published with the Web App name that you provide.</td>
</tr>
<tr>
<td>Supported servers</td>
<td>From the list, select the servers that will be the destination for your published Web App. The primary function of these servers is to support publishing your code for running, testing, and deploying your Web App to PAS for OpenEdge.</td>
</tr>
</tbody>
</table>

---

**PAS for OpenEdge launch configuration settings**

The **Progress Application Server for OpenEdge** launch configuration wizard displays the following tabs:

<table>
<thead>
<tr>
<th>Server</th>
<th>Displays the server name, host name, and runtime associated with the PAS for OpenEdge.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Startup</td>
<td>Lets you specify the session and agent parameters of the PAS for OpenEdge.</td>
</tr>
<tr>
<td>PROPATH</td>
<td>Lets you control where the AVM searches for files and procedures. The PROPATH is associated with the default ABL application in the PAS for OpenEdge instance.</td>
</tr>
</tbody>
</table>
Databases

Let you add the database connection for running the configuration. These database connections are associated with the default ABL application in the PAS for OpenEdge instance.

Common

Let you set options for specifying where the application launch configuration is saved, how it is accessed, and how it runs.

Server tab (PAS for OpenEdge)

The Server tab in the PAS for OpenEdge section of the Configurations wizard specifies the PAS for OpenEdge instance used by the launch configuration.

The following controls are available on the Server tab:

<table>
<thead>
<tr>
<th>Name (accessible from all tabs)</th>
<th>(Required) Specifies the name of the launch configuration, which appears on menus and in the left pane of the Configurations wizard.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>(Required) Specifies the PAS for OpenEdge instance that starts when the launch configuration runs.</td>
</tr>
<tr>
<td>Runtime Environment</td>
<td>(Read-only) Specifies the version of the PAS for OpenEdge instance.</td>
</tr>
<tr>
<td>Host Name</td>
<td>(Read-only) Specifies the machine on which the PAS for OpenEdge broker runs.</td>
</tr>
<tr>
<td>Apply</td>
<td>Saves the current launch configuration definition.</td>
</tr>
<tr>
<td>Revert</td>
<td>Discards unsaved changes to the current launch configuration definition.</td>
</tr>
<tr>
<td>Run or Debug</td>
<td>Starts or debugs the specified PAS for OpenEdge instance.</td>
</tr>
<tr>
<td>Close</td>
<td>Closes the Configurations wizard.</td>
</tr>
</tbody>
</table>

Startup tab (PAS for OpenEdge)

The Startup tab of the Progress Application Server for OpenEdge section of the Configurations wizard specifies the PAS for OpenEdge instance used by the launch configuration.

The following controls are available on the Startup tab:

<table>
<thead>
<tr>
<th>Name (accessible from all tabs)</th>
<th>(Required) Specifies the name of the launch configuration, which appears on menus and in the left pane of the Configurations wizard.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publish directory</td>
<td>Specifies the directory in which the AVM starts when the configuration runs. By default, the value is set to the OpenEdge installation working directory.</td>
</tr>
<tr>
<td></td>
<td>If you created the launch configuration using the New Server wizard, the value you set for the PAS for OpenEdge is used.</td>
</tr>
<tr>
<td></td>
<td>Click Workspace, File System, or Variables to specify a location by browsing or using a variable.</td>
</tr>
</tbody>
</table>
### Session Procedures
Specifies the procedure that is to run when each of these PAS for OpenEdge operations occurs.

Click **Browse** to display and select from the **Select Activation Procedure** dialog.

### Agent Procedures
Specifies the startup and shutdown procedure that is to run when each of these PAS for OpenEdge operations occurs.

### Startup procedure parameters
(Optional) Specifies one or more startup parameters to be used when the startup procedure, if any, runs.

### Agent startup parameters
(Optional) Specifies one or more startup parameters to be used when the server instance starts.

### Import From Server
Loads the **Startup** tab values of the agent properties from `openedge.properties` file.

**Note:** The PROPATH and database information are not copied.

### Apply
Saves the current launch configuration definition.

### Revert
Discards unsaved changes to the current launch configuration definition.

### Run or Debug
Starts or debugs the specified PAS for OpenEdge instance.

### Close
Closes the **Configurations** wizard.

---

### PROPATH tab (PAS for OpenEdge)

The **PROPATH** tab in the **Progress Application Server for OpenEdge** section of the **Configurations** wizard allows you to control where OpenEdge searches for files and procedures when the launch configuration runs.

The **PROPATH** tab provides the following controls:

<table>
<thead>
<tr>
<th>Name (accessible from all tabs)</th>
<th>(Required) Specifies the name of the launch configuration, which appears on menus and in the left pane of the <strong>Configurations</strong> wizard.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tree view</td>
<td>Displays the current PROPATH definition.</td>
</tr>
<tr>
<td>Add Project</td>
<td>Opens a browser that lets you select a project from which the PROPATH settings are imported. OpenEdge copies the PROPATH of the project you choose to the selected location in the launch configuration PROPATH.</td>
</tr>
<tr>
<td>Copy ini Settings</td>
<td>Opens a browser that lets you select an initialization (INI) file from which the PROPATH settings are imported. OpenEdge replaces the entire existing PROPATH with the PROPATH defined in the selected initialization file.</td>
</tr>
<tr>
<td>Other command buttons</td>
<td>Function as described in the PROPATH help for the <strong>Project Properties</strong> dialog.</td>
</tr>
</tbody>
</table>
Agent startup parameters | (Optional) Specifies one or more startup parameters to be used when the server instance starts.
---|---
Import From Server | Loads the **Startup** tab values of the agent properties from openedge.properties file.
| **Note:** The PROPATH and database information are not copied.
Apply | Saves the current launch configuration definition.
Revert | Discards unsaved changes to the current launch configuration definition.
Run or Debug | Starts or debugs the specified PAS for OpenEdge instance.
Close | Closes the **Configurations** wizard.

**Note:** The only OpenEdge substitution variables that this tab supports are **@WORK** and **@DLC**. In addition, the Eclipse substitution variables (**Window > Preferences > Run/Debug > String Substitution**) are supported.

### Databases tab (PAS for OpenEdge)

The **Databases** tab in the **Progress Application Server for OpenEdge** section of the **Configurations** wizard allows you to specify the OpenEdge database connections used by the **Progress Application Server for OpenEdge** launch configuration.

The following controls are available on this tab:

| Name (accessible from all tabs) | (Required) Specifies the name of the launch configuration, which appears on menus and in the left pane of the **Configurations** wizard. |
| Show selected/Show all | Filters the database connection list to show: only those connections that are currently used by the configuration, or all available connections. |
| Database connection list | Lists the available OpenEdge database connections, filtered according to the **Show selected/Show all/Show from project** selection. The launch configuration uses those entries that are checked. |
| Other controls | Function as described in the Database Connections help for the **Project Properties** dialog. |
| Apply | Saves the current launch configuration definition. |
| Revert | Discards unsaved changes to the current launch configuration definition. |
| Run or Debug | Starts or debugs the specified PAS for OpenEdge instance. |
| Close | Closes the **Configurations** wizard. |
PAS for OpenEdge debug configuration settings

The Progress Application Server for OpenEdge debug configuration wizard displays the following tabs:

<table>
<thead>
<tr>
<th>Server</th>
<th>Displays the server name, host name, and runtime associated with the PAS for OpenEdge.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Startup</td>
<td>Lets you specify the session and agent parameters of the PAS for OpenEdge.</td>
</tr>
<tr>
<td>PROPATH</td>
<td>Lets you control where the AVM searches for files and procedures. The PROPATH is associated with the default ABL application in the PAS for OpenEdge instance.</td>
</tr>
<tr>
<td>Databases</td>
<td>Lets you add the database connection for running the configuration. These database connections are associated with the default ABL application in the PAS for OpenEdge instance.</td>
</tr>
<tr>
<td>Source</td>
<td>Lets you add custom source lookup paths to the entries.</td>
</tr>
<tr>
<td>Debugger</td>
<td>Lets you specify connection properties, attach the Debugger to a broker, and add secure client connection properties.</td>
</tr>
<tr>
<td>Common</td>
<td>Lets you set options for specifying where the application launch configuration is saved, how it is accessed, and how it runs.</td>
</tr>
</tbody>
</table>

Migrating existing projects to PAS for OpenEdge

A default PASOE folder that contains the WAR structure is added to the project. If you open 11.2.x, 11.3.x, or 11.4 workspace with OpenEdge 11.5 and if the workspace has any REST or Data Object projects, all the projects are automatically updated with the PASOEContent folder. If you open a 11.5 workspace, you can import the OpenEdge 11.2.x, 11.3.x, or 11.4 projects if the project type is either REST or Data Object, after import the projects are automatically updated with the PASOEContent folder.

New WebHandler Class

The New WebHandler ABL Class wizard helps you create new web handler class file.

This wizard appears when you select File > New > WebHandler.

Note: If WebHandler does not appear in your File menu, click File > New > other and select Progress OpenEdge > Editor > WebHandler. You can alternatively right-click the project and select New > WebHandler.

The following controls are available:

| Package root | Specifies a currently open project to contain the class code and other project code. Click Browse if you want to select a project other than the current one (the default value). |
| Package | Optionally specifies a package name that corresponds to a sub-folder of the package root, and will contain the class file. Click Browse if you want to select a project other than the current one (the default value). A period (.) must separate each subfolder from its parent folder in the path name. |
| WebHandler name | (Required) Specifies the name of the WebHandler class. |
| Final | If selected, specifies that inheritance from this class is not allowed. A FINAL option is included in the generated ABL code. A Final class cannot be Abstract. |
| Abstract | If selected, specifies that the Web handler class is abstract and cannot be instantiated. An abstract class is designed to serve as a super class from which other classes inherit and implement members. Therefore, an Abstract class cannot be Final. |
| Widget pool | If selected, specifies that a USE-WIDGET-POOL option will be included in the generated ABL code. |
| Serializable | If selected, marks to mark the WebHandler class as serializable by including the SERIALIZABLE modifier in the generated ABL code. |
| Note: | If the class inherits from a super class that is not serializable, then it cannot be marked as serializable. |
| Inherits | Optionally specifies another class in the current project as a super class (or a different project that is available in the propath) from which the new class inherits state and behavior in the Inherits field. Click Browse or enter the class path by typing. A period (.) must separate each node from its parent folder in the path name. |
| Note: | By default, the super class name is populated as OpenEdge.Web.WebHandler. The super class cannot be generic. |
| Implements | Lets you optionally specify one or more interfaces in the current project or a different project that is available in the propath that the class implements using the Implements section. Click Add and select the required interfaces in the Interface Selection dialog. Click Remove to remove an interface from the list after adding it. |
| Method stubs | Specifies the methods that you want to generate in the Web handler class. |
Error-handling statement | Adds one of the following error-handling statements to the generated ABL code:
| • Block level: Includes the BLOCK-LEVEL ON ERROR UNDO, THROW statement
| • Routine level: Includes the ROUTINE-LEVEL ON ERROR UNDO, THROW statement

Throw a Not Implemented exception | If selected, specifies that "METHOD NOT IMPLEMENTED" error code is added to stubs for members implemented via interface and stubs for inherited abstract members.

Return a default value | If selected, specifies that the default return values are added to stubs for members implemented through interface and stubs for inherited abstract members.

### Progress OpenEdge Server Monitor view

The Progress OpenEdge Server Monitor view provides the status of and a summary of the properties of a PAS for OpenEdge server instance. You can access the Monitor view as follows:

- From any other perspective, select Window > Show View > Other > Server > Progress OpenEdge Server Monitor
- From the Servers view, right-click anywhere and select Progress OpenEdge Server Monitor

In the Progress OpenEdge Server Monitor view, expand the Progress Application servers for OpenEdge instance. All the deployed web applications and their services are listed. If you select a web application or an ABL REST or Data Object Service, then its properties are listed in the right section. If you select the web node (present under each web application), the deployed web handlers and the resource URI s and the URIs to access the handler classes for the deployed services are listed in the right section.

### Progress OpenEdge Server Monitor view toolbar

You can use the buttons on the Progress OpenEdge Server Monitor view toolbar to accomplish the following tasks:

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Automatic refresh" /></td>
<td>Automatically refresh the server status at a predetermined interval.</td>
</tr>
<tr>
<td><img src="image" alt="Refresh" /></td>
<td>Refresh the server status. (Available only when server is started and selected.)</td>
</tr>
<tr>
<td><img src="image" alt="Add agents" /></td>
<td>Add one or more agents. (Available only when server is started and selected.)</td>
</tr>
<tr>
<td><img src="image" alt="Trim agents" /></td>
<td>Trim one or more agents. (Available only when server is started and selected.)</td>
</tr>
<tr>
<td><img src="image" alt="Trim all agents" /></td>
<td>Trim all agents. (Available only when server is started and selected.)</td>
</tr>
</tbody>
</table>
Open a Web browser by using the currently selected server and start the OpenEdge Explorer connection the server is using.

**Note:** For more information, see Accessing OpenEdge Explorer.

Access Progress OpenEdge Server preferences.

---

### Updating transport properties

When you create an ABL Web App project, by default the merge.template file is available in the `<prj>/PASOEContent/WEB-INF/tlr` location, with the default values for all supported transports. You can update the default values of these transports by editing the values in the `merge.template` file.

For example, if you do not support SOAP clients, you can change the SOAP transport’s `adapterEnabled` property in the `merge.template` file from 1 to 0 as shown below:

```plaintext
# Transport properties for the SOAP protocol
[$ {oepas-app}.${oepas-webapp}.SOAP]
  adapterEnabled=0
  adminEnabled=1
  adminSoapAction=urn:services-progress-com:wsa-admin:01
  debugClients=
  wsaUrl=http://${psc.as.host.name}:${psc.as.http.port}/${oepas-webapp}/soap
  wsd1Enabled=1
```

The modified values (if any) are considered at the time of publishing the application for the first time and exporting the application as a complete Web Application, but not if you are exporting it incrementally.

---

### Using Apache Ant tasks

This section discusses the Ant tasks that you can write for ABL WebApp and Web UI projects to automate processes.

**Note:** For more information about Apache Ant tasks, refer to the Apache Ant documentation.

---

### Packaging an ABL Web App project

You can write an ABLWebAppPackage task to package an ABL Web App project as a .WAR file to publish it as a new and complete Web App or as a .ZIP file to publish the Web App incrementally to a PAS for OpenEdge instance.

**Task-level properties and elements:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>srcdir</td>
<td>Specifies the location of the project for which the .WAR or .ZIP file is to be generated.</td>
</tr>
<tr>
<td>dlc</td>
<td>Specifies the location of your OpenEdge installation.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>verbose</td>
<td>(Optional) Enables the verbose mode. The valid values are true, false, on, off, yes, and no.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> By default, the value is set to false.</td>
</tr>
<tr>
<td>services</td>
<td>(Optional) Specifies the names of the services in the project.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> If there are multiple services, separate the names with a comma.</td>
</tr>
<tr>
<td>isIncremental</td>
<td>(Optional) Specifies whether to generate a .ZIP file for deploying the app incrementally or a .WAR file for deploying the app at once.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> By default, the value is set to false and a .WAR file is generated for full deployment.</td>
</tr>
<tr>
<td>webAppName</td>
<td>(Optional) Specifies a name for the generated .WAR or .ZIP file.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> By default, the name of the project is assigned to the .WAR or .ZIP file.</td>
</tr>
<tr>
<td>destdir</td>
<td>(Optional) Specifies the destination directory where the .ZIP file is generated.</td>
</tr>
</tbody>
</table>

Here is a sample build.xml file for packaging an ABL WebApp project:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project name="ABLWebApp" default="main" basedir="."/>
<property name="dlc.dir" value="C:\Progress\OpenEdge117" />
<property name="dlc.java.dir" value="${dlc.dir}/java" />
<property name="src.dir" value="C:\ablwebapp\src\WebAppPrjRest" />
<property name="dest.dir" value="C:\ablwebapp\dst\WebAppPrjRest" />
<target name="taskdef">
<echo>Task Definitions</echo>
<taskdef resource="com/progress/openedge/pdt/ant/ablwebapp/ablwebapps.properties">
<classpath>
<pathelement location="${dlc.java.dir}/ant-ablwebapp.jar" />
</classpath>
</taskdef>
<target name="ablwebapp" depends="taskdef">
<pathelement location="${dlc.java.dir}/ant-libs/ablwebapp.jar" />
<pathelement location="${dlc.java.dir}/ant-libs/ablwebapp-dependencies.jar" />
</target>
</project>
```

Chapter 4: Overview of Progress Application Server for OpenEdge
Packaging REST services

You can write a PaarGeneration task to package REST services to .PAAR files. These .PAAR files can be deployed to a PAS for OpenEdge instance or exported.

Task-level properties and elements:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>srcdir</td>
<td>Specifies the location of the project for which .PAAR files are to be generated.</td>
</tr>
<tr>
<td>dlc</td>
<td>Specifies the location of your OpenEdge installation.</td>
</tr>
<tr>
<td>verbose</td>
<td>(Optional) Enables the verbose mode. The valid values are true, false, on, off, yes, and no.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> By default, the value is set to false.</td>
</tr>
<tr>
<td>services</td>
<td>(Optional) Specifies the names of the services available in the project.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> If there are multiple services, separate the names with a comma. If you do not specify a service, .PAAR files are generated for all the REST services in the project.</td>
</tr>
<tr>
<td>destdir</td>
<td>(Optional) Specifies the destination directory where the .PAAR files are generated.</td>
</tr>
</tbody>
</table>

Here is a sample build.xml file for packaging a REST service:

```xml
<?xml version="1.0" encoding="UTF-8"?><project name="ABLWebApp" default="main" basedir=".">

<property name="dlc.dir" value="C:\Progress\OpenEdge117" />
<property name="dlc.java.dir" value="${dlc.dir}/java" />
<property name="app.name" value="ExportWebApp1" />
<property name="src.dir" value="C:\ablwebapp\src\${app.name}" />
<property name="dest.dir" value="C:\ablwebapp\dst\ant\paars" />

<!-- Main task -->
<target name="main" depends="taskdef">
<ABLWebAppPackage srcdir="${src.dir}" dlc="${dlc.dir}" verbose="false" isIncremental="true" webAppName="${app.name}" destdir="${dest.dir}" />
</target>
</project>
```
<echo>Task Definitions</echo>

<taskdef resource="com/progress/openedge/pdt/ant/ablwebapp/ablwebapps.properties">

<classpath>

<pathelement location="${dlc.java.dir}/ant-ablwebapp.jar" />
<pathelement location="${dlc.java.dir}/ant-libs/ablwebapp.jar" />
<pathelement location="${dlc.java.dir}/ant-libs/ablwebapp-dependencies.jar" />

<!-- CodeModel Dependencies -->
<pathelement location="${dlc.java.dir}/ant-libs/codemodel-dependencies.jar" />

<!-- AST and its Dependencies -->
<pathelement location="${dlc.java.dir}/ant-libs/ast.jar" />
<pathelement location="${dlc.java.dir}/ant-libs/ast-dependencies.jar" />

<!-- Additional deps -->
<pathelement location="${dlc.java.dir}/ant-libs/velocity-1.7.jar" />
<pathelement location="${dlc.java.dir}/ant-libs/velocity-1.7-dep.jar" />
<pathelement location="${dlc.java.dir}/ant-libs/lpadapters-restExpose.jar" />
<pathelement location="${dlc.java.dir}/lpadapters-idl.jar" />
<pathelement location="${dlc.java.dir}/ext/jettison-1.2.jar" />

</classpath>
</taskdef>
</target>

<!-- Main task -->
<target name="main" depends="taskdef">

<PaarGeneration srcdir="${src.dir}" dlc="${dlc.dir}" verbose="false" destdir="${dest.dir}"${app.name}" />

</target>

</project>

Generating a Data Object Service Catalog file

You can write a CatalogGeneration task to generate a Catalog file for a Data Object Service in an ABL Web App project.

Task-level properties and elements:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>srcdir</td>
<td>Specifies the location of the ABL Web App project.</td>
</tr>
<tr>
<td>dlc</td>
<td>Specifies the location of your OpenEdge installation.</td>
</tr>
<tr>
<td>verbose</td>
<td>(Optional) Enables the verbose mode. The valid values are true, false, on, off, yes, and no.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> By default, the value is set to false.</td>
</tr>
<tr>
<td>serviceName</td>
<td>Specifies the name of the service.</td>
</tr>
</tbody>
</table>
Specifies a comma-separated list of resources (classes and procedures) associated with the Data Object Service.

**Note:** Make sure to specify at least one resource and the resources must be accurately annotated.

(Optional) Specifies the destination directory where the Catalog file is generated.

**resources element**

Contains a set of ABL source files that are required to generate the Catalog file. You can configure this element using pathelement or fileset elements.

**Note:** For more information on pathelement or fileset elements, refer to the Apache Ant documentation.

**dbinfo element**

Contains the database connection details that the catalog generation task depends on. Each dbinfo element is considered a single database connection information.

**Note:** Make sure that the server and database is running when you run the ANT task.

Here is a sample build.xml file for Data Object Service catalog generation:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project name="ABLWebApp" default="main" basedir="."/>

<property name="dic.dir" value="C:\Progress\OpenEdge117" />
<property name="dic.java.dir" value="${dic.dir}/java" />
<property name="src.dir" value="C:\ablwebapp\src\ABLWebAppAllSrvs" />
<property name="dest.dir" value="C:\ablwebapp\dst\ABLWebAppAllSrvs" />
<property name="src.dir" value="C:\ablwebapp\src\CatalogAnt\test" />
<property name="dest.dir" value="C:\ablwebapp\src\CatalogAnt\cat" />
<property name="build.dir" value="C:\OfficeWork\streams\psajja_OE_117_pdsoe\vobs_oeide\nt" />
<property name="dlc.home" value="${build.dir}/dlc/" />
```
<!-- Copy the jars from latest build -->
<target name="copy">
<copy todir="${dlc.java.dir}/ant-libs" overwrite="true" />
</copy>
<copy file="${build.dir}/ant_webapps/dist/ant-ablwebapp.jar" todir="${dlc.java.dir}" overwrite="true" />
</target>

<!-- Target for defining 'taskdef' -->
<target name="taskdef" depends="copy">
<echo>Task Definitions</echo>
<taskdef resource="com/progress/openedge/pdt/ant/ablwebapp/ablwebapps.properties">
<classpath>
<pathelement location="${dlc.java.dir}/ant-ablwebapp.jar" />
</classpath>
</taskdef>
</target>

<!-- Copy task -->
<target name="copytsk">
<copy todir="${dlc.home}/oeide">
<fileset dir="${build.dir}"/>
<include name="Architect_repo/plugins/com.progress.openedge.pdt.ablwebapp_*" />
<include name="eclipse/**" />
</fileset>
</copy>
</target>

<!-- Main task -->
<target name="main" depends="taskdef">
<mkdir dir="${dest.dir}"/>
<CatalogGeneration srcdir="${src.dir}" dlc="${dlc.dir}" verbose="true" resources="${src.dir}\AppServer\tst.cls" serviceName="dorest" destdir="${dest.dir}" />
</CatalogGeneration>

<!-- Configure Database Info -->
<dbinfo name="sports2000" host="localhost" port="4546"/>
</CatalogGeneration>
Packaging a Web UI project

You can write a WebUiPackage task to package a Web UI project as a .WAR file to publish it as a new Web App to a PAS for OpenEdge instance.

**Task-level properties and elements:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>srcdir</td>
<td>Specifies the location of the project for which .WAR file is to be generated.</td>
</tr>
<tr>
<td>dlc</td>
<td>Specifies the location of your OpenEdge installation.</td>
</tr>
<tr>
<td>verbose</td>
<td>(Optional) Enables the verbose mode. The valid values are true, false, on, off, yes, and no.</td>
</tr>
<tr>
<td>buildType</td>
<td>(Optional) Specifies if the project is for publishing on a production build or debugging on a development build. The valid values are release and debug.</td>
</tr>
<tr>
<td>webAppName</td>
<td>(Optional) Specifies a name for the generated .WAR file.</td>
</tr>
<tr>
<td>destdir</td>
<td>(Optional) Specifies the destination directory where the .WAR file is generated.</td>
</tr>
</tbody>
</table>

**Note:** By default, the value is set to false.

Here is a sample build.xml file for packaging a Web UI project:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project name="ABLWebApp" default="main" basedir="."/>

<property name="dlc.dir" value="C:\Progress\OpenEdge117" /> 
<property name="dlc.java.dir" value="${dlc.dir}/java" />
<property name="app.name" value="WebUiPrj" />
<property name="src.dir" value="C:\ablwebapp\src\${app.name}" />
<property name="dest.dir" value="C:\ablwebapp\dst\ant\wars1\${app.name}" />

<!-- Target for defining 'taskdef' -->
<target name="taskdef" depends="copy">
  <echo>Task Definitions</echo>
  <taskdef resource="com/progress/openedge/pdt/ant/ablwebapp/ablwebapps.properties">
    <classpath>
      <pathelement location="${dlc.java.dir}/ant-ablwebapp.jar" />
      <pathelement location="${dlc.java.dir}/ant-libs/ablwebapp.jar" />
      <pathelement location="${dlc.java.dir}/ant-libs/ablwebapp-dependencies.jar"
    </classpath>
  </taskdef>
</target>
```
<!-- CodeModel Dependencies -->
<pathelement location="${dlc.java.dir}/ant-libs/codemodel-dependencies.jar" />

<!-- AST and its Dependencies -->
<pathelement location="${dlc.java.dir}/ant-libs/ast.jar" />
<pathelement location="${dlc.java.dir}/ant-libs/ast-dependencies.jar" />
<pathelement location="${dlc.java.dir}/ext/jettison-1.2.jar" />

</classpath>
</taskdef>
</target>

<!-- Main task -->
<target name="main" depends="taskdef">
  <WebUiPackage srcdir="${src.dir}" dlc="${dlc.dir}" verbose="false"
    webAppName="${app.name}" destdir="${dest.dir}" buildType="release" />
</target>
</project>
Introducing OpenEdge REST

OpenEdge provides the ability for REST clients to initiate the execution of business logic on an OpenEdge AppServer, and to receive back results from that execution. REST (Representational State Transfer) is an architectural style for distributed computing that was defined in conjunction with the HTTP protocol. A REST client is similar to any other OpenEdge client, and uses HTTP as its communication protocol.

For details, see the following topics:

- Concepts
- Tasks
- Reference

Concepts

Project support for REST

Most operations that you perform with Progress Developer Studio for OpenEdge take place within the context of a project. You can create multiple REST projects and maintain them concurrently in your workspace. By default, the REST project opens in OpenEdge Server perspective.

**Note:** REST projects can be deployed only to traditional servers not PAS for OpenEdge.

You can create a REST project and a REST module. For details, see Creating a REST project on page 208.
REST project type

A REST project type is an OpenEdge project that is specialized for developing and deploying ABL REST Web applications. An OpenEdge REST project contains the following OpenEdge facets:

- REST
- AppServer
- Progress Adapters

You can see the facets enabled for your project using the Project Facets page.

See also

Creating a REST project on page 208
Developing ABL REST Web applications on page 210
Deploying ABL REST Web applications on page 219
REST Expose Editor on page 228

Terminology related to REST support

The following terms describe various components that apply to REST support in Progress Developer Studio for OpenEdge:

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABL routine</td>
<td>ABL routines are internal procedures, user-defined functions, or class methods.</td>
</tr>
<tr>
<td>REST service</td>
<td>A service defined to expose your REST interface annotated ABL class and procedure files to a Web server as a REST Web application WAR file. The defined REST service appears under the Defined Services node in the project directory. For more information, see Defining a REST service on page 211.</td>
</tr>
<tr>
<td>REST Web application</td>
<td>A WAR file that includes one or more REST services. Any REST Web application, using its WAR file, is an independently deployable and manageable Web server application. The REST Web application will be deployed on the Apache Tomcat Web applications directory at $DLC/servers/tomcat/webapps. You can publish a REST service associated with an OE Web Server instance to create a REST Web application WAR file. To package multiple REST services into a single REST Web application WAR application, use the Exporting a REST Web application on page 222 wizard.</td>
</tr>
<tr>
<td>REST resource</td>
<td>A resource is any URI that resolves to a single HTTP method call. For more information, see Adding REST resources on page 214.</td>
</tr>
</tbody>
</table>
A Java container provided with OpenEdge that allows you to deploy the REST services and REST Web applications. When installing Progress Developer Studio for OpenEdge, Apache Tomcat is also installed and configured. The Tomcat is located in your OpenEdge installation directory at $DLC/servers/tomcat.

OE Web Server
A Web server that enables you to deploy, configure, and manage REST services/applications to the Java container. It helps you to collect statistics and provide runtime configuration for a REST Web application. For more information, see Creating an OE Web Server instance on page 220.

See also
Generated artifacts for a REST service on page 205
Creating a REST project on page 208

REST service artifacts
The REST services you define using the New REST service wizard appears under the Defined Services node in your REST project.

The REST service artifacts are generated under .services/Expose/rest/service_name in your project directory, where service_name is the name of the defined REST service. A separate REST service node is created for each REST service you define, and its artifacts are generated under that REST service.

Note: The .services node does not appear by default under your project in the Project Explorer view. To view the .services node, open Available Customizations dialog (select View Menu > Customize Menu from the Project View toolbar), clear the *.resources check box, and then click OK.

Each of the REST service node includes the following two files:

- resourceModel.xml - Includes the resource URI information.
- spring.xml - Includes the AppServer end point details, resourceModel.xml, and mapping.xml entry.

Note: Progress recommend you to not modify or delete any file under the .services node in your REST project.

See also
Defining REST service on page 211
Working with REST Expose Editor on page 212
Creating REST project on page 208

REST service annotations
The REST service annotations falls into two categories: REST main annotations and REST detail annotations.
REST main annotations

REST main annotation makes an ABL procedure or class, or an external procedure file available as a REST resource. This annotation must precede any ABL statement in the procedure or class file.

The illustration below provides an example of REST main annotation:

```
@openapi.openedge.export FILE(type="REST", executionMode="{external | single-run | singleton}", useReturnValue="{true | false}", writeDataSetBeforeImage="{true | false}").
```

File

This keyword specifies that the annotation applies to an ABL class or procedure level. This is a mandatory attribute for main annotation.

type

This attribute specifies the type of defined service such as REST for a REST service or ESB for an ESB service. This is a mandatory attribute.

executionMode

This attribute specifies how the ABL class or procedure, or an external procedure runs. If you specify the value as EXTERNAL, the procedure runs externally, the value SINGLE-RUN allows you to call internal procedures and user defined functions in a procedure file, and the value SINGLETON allows you to call methods in a class file or internal procedures in a procedure file. This is a mandatory attribute.

useReturnValue

This attribute specifies whether a return string is used for procedures. The default value is false. This is an optional attribute.

writeDataSetBeforeImage

This attribute specifies whether before-image data is written out when ProDataSet parameters are serialized as XML. Before-image is not supported for JSON. The only valid value for REST is false. This is an optional attribute.

See also
Defining REST service annotations on page 210
REST detail annotations on page 206

REST detail annotations

REST detail annotation makes an ABL routine (internal procedures, methods, and user-defined functions) available as a REST resource. This annotation must precede the internal procedure or function, or method statements. When you use this annotation, you must also use an REST main annotation at the beginning of the source files.

```
@openapi.openedge.export(type="REST", useReturnValue="{true | false}", writeDataSetBeforeImage="{true | false}").
```
type

This attribute specifies the type of defined service such as REST for a REST service or ESB for an ESB service. This is a mandatory attribute.

useReturnValue

This attribute specifies whether a return string is used for the internal procedures. The default value is false. This is an optional attribute.

writeDataSetBeforeImage

This attribute specifies whether before-image data is written out when ProDataSet parameters are serialized as XML. The default value is false. If no ProDataSer parameters exist, writeDataSetBeforeImage is ignored. Before-image data is not supported for JSON. This is an optional attribute.

See also
Defining REST service annotations on page 210
REST main annotations on page 206

Default parameter mapping

REST supports default parameter mapping. The default parameters mapping is supported for only the query and path parameters in an HTTP request.

The default parameter mapping for REST will follow these rules:

Note: Input parameter mean either Input or Input-Output.

• If an input parameter name is contained in the resource URI as a query, then the input parameter is automatically mapped as a query parameter. For example, if a query parameter ?custid=[custid] is contained in a resource URI and an input parameter custid is contained in the interface parameters, then the query parameter is automatically mapped with that input parameter.

• If an input parameter name is contained in the resource URI as a path, then the input parameter is automatically mapped as a query parameter. For example, if a path parameter {custid} is contained in a resource URI and an input parameter custid is contained in the interface parameters, then the path parameter is automatically mapped with that input parameter.

See also
Associating REST resource verbs with operations on page 216
Mapping REST resource verb parameters on page 217
Chapter 5: Introducing OpenEdge REST

Tasks

Building a REST Web application

To expose an ABL class or procedure file as a REST service, follow these steps:

1. Create an OpenEdge REST project using the New OpenEdge Project wizard.
2. Annotate ABL procedure and class files with REST interface annotations using the Define Service Interface wizard.
3. Create REST service using the New REST Service wizard.

**Note:** If you select to create a default REST service when creating the REST project, you can skip this step.

4. Map REST resource verbs and parameters using the REST Expose Editor.
5. Package as a REST Web application WAR file using the Export REST Application wizard.
6. Deploy REST services to the Apache Tomcat Java container using the Publish option.

See also

- Developing REST services on page 210
- Deploying REST services on page 219
- Packaging as a REST Web application WAR file on page 222

Creating a REST project

To create an OpenEdge project specialized for REST Web application development:

1. Select File > New > OpenEdge Project from the Progress Developer Studio for OpenEdge main menu bar or the Project Explorer view context (right-click) menu. The New OpenEdge Project wizard opens and displays the Create an OpenEdge Project page on page 105.
2. Type a name in the Project name field.

   The root folder for the new project is created in your current Eclipse workspace, unless you specify a different location.

3. To choose a different location for the project, click and clear the Use default check box. Then either browse to, or enter the path to the desired location.
4. Select Server from the left section.
5. Select Classic option button to select the server type that you want to deploy the application to.
6. Select REST as the transport.
7. Click Next. The Select AVM and layout options page appears.
8. Specify the OpenEdge AVM options and the project layout options (or leave the default settings) and click Next. The Define AppServer content module page appears.
9. In the **AppServer source folder** field, enter a name of the AppServer module content directory.

   **Note**: The default is a folder named AppServer, which will be created as a child of the project folder if you select the **Use project root directory for source and r-code** check box in step 6. If you select the **Use separate source and r-code directories** option, the folder will be a child of whatever you specified in the **Source directory** field.

10. From the **Supported servers** list, select the servers that will be the destination for your published web objects. The primary function of these servers is to support publishing your code for running, testing, and deploying your REST service. For example, select **restbroker1** from the list to publish your REST service.

11. Select the **Create a REST service** to define a default REST service and associate it to an OE Web Server instance. The default name of the REST service is the REST project name and Service appended to it. For example, `<projectname>`Service. You can specify a different name in the **Service name** field and a relative URI in the **Service relative URI** field.

12. Click **Next**. The **Define PROPATH** page appears.

13. Leave the default settings or modify the PROPATH entries to be used by the project and click **Next**. The **Select database connections** page appears.

14. Select the database connections to be used by the project and click **Finish**.

You will see your project and its resources in the **Project Explorer** view.

   **Note**: A default PASOE folder that contains the WAR structure is added to the project. If you open 11.2.x, 11.3.x, or 11.4 workspace with OpenEdge 11.5 and if the workspace has any REST or Data Object projects, all the projects are automatically updated with the PASOEContent folder. If you open a 11.5 workspace, you can import the OpenEdge 11.2.x, 11.3.x, or 11.4 projects if the project type is either REST or Data Object, after import the projects are automatically updated with the PASOEContent folder.

---

**See also**

[Defining a REST service](#) on page 211

---

### Creating a REST project with project facets

You can use the **Project Facet** properties page to associate the REST facet to an existing OpenEdge project (non-REST project). This changes the nature and behavior of the selected OpenEdge project as a REST project. The AppServer and REST modules will be associated with the project.

To create a REST project with project facets:

1. On the **Project Explorer** view, select an OpenEdge project.

2. Right-click the project, and then select **Properties** on the context menu. The **Properties** page of the selected OpenEdge project appears.

3. Select **Project Facets**. The **Project Facets** page appears.

4. From the **Project Facet** list, select the following facets:
   - **AppServer**
   - **Progress Adapters**
5. Click OK to save your settings and close the properties page.

On the Project Explorer view, you can view the following REST related modules under the project in the Project Explorer view:

- .services
- AppServer
- RESTContent

See also
Creating a REST project on page 208

Developing ABL REST Web applications

Defining REST service annotations

To create an REST Web application, you must identify the procedures and classes to be exposed as REST services. You must add REST interface annotations to the ABL code that specify which procedures or classes should be exported. To use a procedure .p or class .cls file, you must add the service interface definition to it. Defining the service interface refers to defining or adding REST interface annotations in the ABL class and procedure file.

1. Right-click on the ABL class or procedure file in the Project Explorer view, Source, or the Procedure Editor tab, and select Progress OpenEdge > Define Service Interface on the context menu. The Define Service Interface wizard appears.

2. Select REST from the Definition Mode drop-down.

3. In the Project Explorer view, the ABL class or procedure selected in step 1 is automatically checked from the list of procedures and classes in the workspace, and the corresponding resources, internal functions, procedures, and methods are listed in the ABL routines area.

4. In the Backup area, select the Backup selected resources check box to back up the procedure files before defining the service interface.

5. In the Location area, click Browse and specify the backup location, and then click Next. The Edit Annotation page appears. It displays the following parameters in the REST main annotation and REST detail annotation sections:

   - **File** - This attribute is mandatory. It specifies that the annotation applies to the main annotation at the ABL class or procedure level.
   - **Type** - This attribute is mandatory. The value of this attribute is a comma separated list of types. If you are annotating an external procedure for use with an AppServer REST Adapter, the value of this attribute must include REST. If you are exporting a procedure for use with both the REST Adapter and the ESB Adapter, then the value of this attribute can include both types.
### Tasks

- **executionMode** - This attribute is mandatory. It specifies how the ABL class or procedure, or an external procedure runs. If you specify the value as `EXTERNAL`, the procedure runs externally, the value `SINGLE-RUN` allows you to call internal procedures and user defined functions in a procedure file, and the value `SINGLETON` allows you to call methods in a class file or internal procedures in a procedure file. Ensure that the top-level procedure contains no parameters and the state is discarded between calls to internal procedures and functions. Persistent procedures are not supported since REST services are stateless in nature.

- **useReturnValue** - This attribute is optional. It specifies whether a return string is used for procedures. The default value is false.

- **writeDataSetBeforeImage** - This attribute is optional and the only valid value for REST is false. It specifies whether before-image data is written out when ProDataSet parameters are serialized as XML. Before-image is not supported for JSON.

6. In the **REST main annotation** area, click **Edit** to edit the REST annotation for the ABL file. The **New REST Annotation** dialog appears.

7. In the **New REST Annotation** dialog, you can specify the following:
   a) Enter the type of annotation to be added in the **Type** field.
   b) Select **EXECUTION MODE** from the drop-down.
   c) Select **USE RETURN VALUE** check box, if the ABL procedure file returns a value. A `retVal` parameter is added to the procedure file to store the return value.
   d) Select **WRITE DATASET BEFORE IMAGE** check box, if you want before-image data written to the OpenEdge file when ProDataSet parameters are serialized as XML.

8. Click **OK**.

9. In the **REST detail annotation** area, click **Edit** to edit the REST annotation for the ABL routine of the main procedure file. The **New REST Annotation** dialog appears, perform the following steps:
   a) Select **USE RETURN VALUE** check box, if the selected internal procedure returns a value. A `retVal` parameter is added to the procedure file to store the return value.
   b) Select **WRITE DATASET BEFORE IMAGE** check box, if you want before-image data written to the OpenEdge file when ProDataSet parameters are serialized as XML.

   **Note**: You can edit REST detail annotation only if an ABL routine is selected in 3 on page 210.

10. Click **Finish**.

**See also**

Developing ABL REST services on page 210

### Defining a REST service

The **New REST Service** dialog allows you to define a REST service. You can use the defined REST service to expose your REST interface annotated ABL class and procedure file as a REST Web application.

To define a REST service:
1. From the **Project Explorer** view, select a project for which you want to create a REST service.
2. Right-click the project, and then select **New > REST Service** on the context menu.

   **Note:** If you do not see **REST Service** on the context menu, then select **Other**. On the **Select a wizard** dialog, select **Progress OpenEdge > Service > REST Service**, and then click **Next**.

The **New REST Service** dialog appears.

3. In the **Service name** field, enter a valid name for the REST service. This is a mandatory field.

   **Note:** You cannot have REST services with the same name under the same OpenEdge project.

4. In the **Service description** field, enter a description for the REST service. This is an optional field.

5. In the **Service relative URI** field, enter a relative URI for all the resources to be defined in the REST service. The relative path must start with "/". For example, /service relative URI. This is a mandatory field.

6. In the **Project** field, select a project where you want to generate the REST service. By default, the field displays the project you have selected in step 1 on page 212. You can browse and select a different project. This is a mandatory field.

7. From the **OE Web Servers** list, select the OE Web Server instances to which you want to associate the defined REST service.

   **Note:** This lists the OE Web Server instances of your current workspace, you define using the **New Servers** wizard.

8. Click **Finish**.

This creates the REST service under the **Defined Services** node in the selected REST project, and then opens the **REST Expose Editor**.

**Note:** When you define a REST service for a non-REST project such as AppServer project, the REST and Progress Adapters facets are enabled by default for that project. The project behaves as a REST project enabling you to develop and deploy REST Web applications. The **Project Facets** page displays the facets associated for your current OpenEdge project. See **Projects Facets** page.

**See also**

- REST service artifacts on page 205
- Using the REST Expose Editor on page 213
- REST Expose Editor on page 228
- Packaging REST service as a REST Web application on page 222

**Working with the REST Expose Editor**

**Opening the REST Expose Editor**

The **REST Expose Editor** allows you to define a mapping between the ABL class and procedure file with REST interface annotations and the defined REST service. You can manage the REST resources defined for the REST service, associate REST resource verbs (HTTP methods) with the REST interface annotated ABL routines, and map the input and output parameters.
The **REST Expose Editor** opens by default on creation of a REST service. However, you can reopen the REST editor if you have closed the editor while working or want to view or modify the mappings of the defined REST Web application.

To open the REST Expose Editor:

1. From the **Project Explorer** view, expand the OpenEdge project where you have defined a REST Web application.
2. Expand the **Defined Services** node.
3. Right-click the REST service that you want to open, and then select **Edit** from the context menu.

The selected REST service opens in the **REST Expose Editor**.

**See also**
- Using the **REST Expose Editor** on page 213
- REST Expose Editor on page 228

**Using the REST Expose Editor**

The **REST Expose Editor** allows you to define mapping between the exposed ABL applications (ABL class and procedure files with REST interface annotations) and the defined REST service.

Mapping the defined REST service with the exposed ABL class and procedure files include the following steps:

1. Adding REST resources to the defined REST service. For more information, see Adding REST resources on page 214.
2. Mapping REST (HTTP) verbs with an REST interface-annotated ABL class (.cls) and procedure (.p) files. For more information, see Associating REST verbs with operations on page 216.
3. Mapping the HTTP Request and Response parameters with the Interface parameters. For more information, see Mapping REST resource verb parameters on page 217.

**See also**
- Defining a REST Web application on page 211
- Working with the **REST Expose Editor** on page 212
- REST Expose Editor on page 228

**Modifying service relative URI**

You can use the **Edit Service Relative URI** dialog on the **REST Expose Editor** to modify the relative URI of a defined REST service.

To edit the application relative path:

1. Open the REST service of which you want to modify the relative URI with the **REST Expose Editor**.
2. On the **REST Expose Editor**, click **Edit** beside the **Service relative URI** field. The **Edit Service Relative URI** dialog appears.
3. In the **Service relative URI** field, enter a different relative URI for the REST service.
Note: This is the relative path for all the resources defined in the REST service. The relative URI must start with "/". For example, "/<service resource path>".

4. Click OK to save your changes.

The modified relative path appears in the Service relative URI field on the REST Expose Editor.

See also
Opening the REST Expose Editor on page 212
Edit Application Relative Path dialog on page 229
REST Expose Editor on page 228

Adding REST resources

The New REST Resource dialog allows you to add REST resources to a defined REST service.

Note: Each REST resource you create includes these four REST verbs (HTTP methods): GET, PUT, POST, and DELETE. These REST verbs appear under Verb Association on the REST Expose Editor.

To add a REST resource:

1. On the Resources toolbar, click the Add Resource icon.
   The New REST Resource dialog appears.

2. In the Resource URI field, enter a Uniform Resource Indentifier (URI) for the resource. The REST resources you add are identified with Uniform Resource Identifier (URI). The name must start with "/". For example, /resource URI.

   Note: In a REST resource URI, you cannot use the Java reserved words such as int, char, or boolean, as the path or query parameter names. For more information, see Adding REST parameters on page 214.

3. Click OK.

This creates the REST resource and displays on the Resources list.

Note: Any special characters specified in the resource URI must be encoded while constructing the URI, as per the HTML URL Encoding standards. For more information on constructing a resource URI, see the OpenEdge Application Server - Administration guide.

See also
Opening REST Service with REST Expose Editor on page 212
Adding REST parameters on page 214
Deleting a REST resource on page 216
Modifying a REST resource name on page 215

Adding REST parameters

The REST service URI supports query, path, and form parameters.
Note: You cannot use the Java reserved words as the parameters such as int, char, or boolean.

**Path parameter**

Path parameters are used to identify entities, represent hierarchies or to provide data to a resource. They are not intended to filter or restrict information like query parameters. A resource can have more than one path parameter to represent hierarchical data. The curly braces are only used to identify path parameter values.

The illustration below provides URI examples with path parameters:

```
/customers/{customer-id}, where {customer-id} indicates it is a path parameter, and customer-id is the path parameter value.
```

```
/customers/{customer-id}/orders/{order-id}, where customer-id and order-id are the two path parameters defined in the URI.
```

You can add a path parameter using the New REST Resource dialog on page 229

**Query parameter**

Query parameters represents search criterion and restricts the result list like a filter. They are not intended to identify entities of a resource or to select the representation like path parameters. A query parameter starts with a "?". A resource can have more than one query parameter that is separated with ";".

The illustration below provides URI examples with query parameter:

```
/customers/{customer-name}/info?zip=[cust-zip], where ? indicates that it is a query parameter, zip is the parameter name, and [cust-zip] is the query value.
```

```
/customers/{customer-name}/info?zip=cust-zip&name=cust-name, where zip and cust-name are the two query parameters defined in the URI.
```

Here the customer zip code is passed as a query parameter using which the query returns the matching customer names.

Note: In a resource URI, you must always specify a query parameter after a path parameter. Any path parameter specified after a query parameter will be handled as a query parameter.

You can add a query parameter using either the New REST Resource dialog on page 229 or Managing nodes on page 218

**Form parameter**

Form parameters can be used with only the PUT and POST verbs. You can add a form parameter using the Add Node dialog.

See also

REST Expose Editor on page 228

### Changing a REST resource name

The Edit Resource option allows you to modify the name of a selected REST resource.

To modify the REST resource name:

1. On the Resources list, right-click a REST resource for which you want to modify the resource name, and then select Rename Resource from the context menu.

   The selected REST resource is highlighted for inline editing.

2. Enter a different resource name, and then press ENTER.

This modifies the name of the selected REST resource.
See also
Opening the REST Expose Editor on page 212
Adding a REST resource on page 214
Using the REST Expose Editor on page 213

Deleting a REST resource
You can use the Delete Resource option to delete a REST resource that you have added to a REST service.

Note: When you delete a REST resource, the associated REST verbs and its mappings are deleted.

To delete a REST resource:
1. In the Resources list, select a REST resource that you want to delete.
2. Do one of the following:
   • On the Resource list toolbar, click the Delete resource icon.
   • Right-click the REST resource, and then select Delete resource from the context menu.

Note: If you delete a parent REST resource, all its child resources are deleted.

3. Click Yes on the confirmation prompt to delete the selected resource.

This removes the selected REST resource from the Resources list.

See also
Opening the REST Expose Editor on page 212
Adding a REST resource on page 214
Using the REST Expose Editor on page 213

Associating REST verbs with operations
The Associate Operation with Verb dialog allows you to associate an operation (REST interface-annotated ABL procedure or class file) with a verb of a REST resource.

All the REST interface-annotated ABL class and procedure files available in your current project are displayed under the Resources list on the Associate Operation With Verb dialog. You can associate each REST resource verb with only one operation.

Note: You cannot associate a REST verb with multiple operations.

To associate an operation with a verb:

1. On the REST Expose Editor, click the Select operation icon beside a verb of a REST resource. For example, From the Verb Association section, click the Select operation icon beside the GET verb.
The **Associate Operation With Verb** dialog appears with a list of REST interface-annotated ABL class and procedure files.

2. From the **Resources** list, select an operation which you want to associate with the selected verb. If the selected ABL class or procedure file includes an exposed internal procedure, methods, or UDFs, they appear under the **ABL routines** list.

3. From the **ABL routines** list, select an REST interface-annotated ABL routine that you want to associate with the verb.

4. Click **OK**.

The selected operation appears beside the associated verb in the **Verb Association** section.

**See also**
- [Associate Operation With Verb dialog](#) on page 230
- [Working with the REST Expose Editor](#) on page 212

### Mapping REST resource verb parameters

After you specify mapping between the REST interface-annotated ABL routine and the defined REST resource verb, you need to create mapping rules that associate the interface parameters (input and output parameters of an ABL routine) with the response or request parameters (input and output parameters in the REST resource).

The **REST Expose Editor** provides a visual mapping tool to create the mapping rules.

A REST resource verb includes two distinct parameter mapping:

- **Request Mapping** - Interface input and output parameters can be mapped to any of the header, body, cookie, query string parameters, path parameter, or form parameter elements of the incoming HTTP request.
- **Response Mapping** - Interface Input-Output and Output parameters are mapped to any combination of header, body, cookies and response code elements in the outgoing HTTP response.

When you select a REST resource verb in the **Verbs Association** section, the mapping definition section on the **REST Expose Editor** displays hierarchical representations of the REST resource parameters and the ABL routine parameters. There is a tab for each type of parameter mapping: **Input** and **Output**.

On **Input** tab, input parameters to operation are mapped from elements of a request component that consists of the following:

- Request URL parameters including resource URI, query string parameter, and form parameter
- Request HTTP message including method, cookies, and headers
- Server context including servlet request, response, context, and config

You can map each element of the request to input parameters.

On **Output** tab, output parameters from an operation are mapped to a HTTP (REST resource) response.

Elements of a HTTP parameters include:

- Interface parameters
- Advanced parameters including constants

Elements of a HTTP response include:

- Response status line including response code
- Response headers (also content-type) including headers and cookies
- Response body
In the mapping definition section, the source tree is on the left and the target on the right. For input parameters, the Request is the source and the interface parameters (ABL routine parameters) is the target; for output parameters the interface parameters (ABL routine parameters) is the source and the Response is the target. To create a mapping rule, click on the parameter field and drag a line to the required operation parameter.

When creating a rule for mapping an interface parameters that contains a simple type (integer, string, or boolean) to a HTTP (REST resource) parameter that contains similar type, you can drag a line between the elements as follows:

- Click on an element that you want to map and drag a line to an element you want to map it to. Repeat this step for each mapping rule that want to define.

The mapping section represents each defined rule with a blue line between elements.

**Note:** You can delete a defined mapping rule, by selecting the blue line defined between the elements and selecting **Delete** from the context menu or pressing the **DELETE** key. **Delete All** option on the context menu allows you to remove all the defined mapping rules.

---

**See also**

- Default parameter mappings on page 207
- Working with REST Expose Editor on page 212
- Adding nodes on page 218

---

**Managing nodes**

The **Mapping Definitions** section on the **REST Expose Editor** allows you to add, modify, or delete the custom node entries added to an element of a HTTP request or response.

**Note:** The option of adding a node is available for only for HTTP Header and it's children.

To add a node:

1. Select an element of a HTTP request or response.
2. Right-click the element, and then select **Add node** on the right-click context menu.
   
   The **Add Node** dialog appears.
3. From the **Type** down-down, specify an element type for which you want to create the node such as Query String Parameter, Header, Body, or Cookie. If you select a PUT or POST verb on the **Verb Association** section, the **Type** drop-down will also include the Form Parameter type.

**Note:**

For a HTTP response, you can see only Header, Body, and Cookie in the **Type** drop-down.

If you selected Header as Type, you can select content-type as the Header name from the drop-down list.

4. In the **Query String Parameter**, **Form Parameter**, **Header**, or **Cookie** field, specify a value for the selected element. For example, `Query1` for Query String Parameter. The **Expression value** field displays the specified parameter value. For example, `${rest.queryparam['Query1']}`.

   The following example illustrates how to specify the **Cookie** field (a **CHAR** or **LONGCHAR** type). After choosing the Cookie field from the drop-down list, you set the cookie (as per the HTTP cookie standard) as follows:
Set-Cookie: HSID=AQYQVn.DKrdst; Domain=.foo.com; Path=/; Expires=Wed, 13-Jan-2021 22:23:01 GMT; HttpOnly

Set-Cookie is a directive for the browsers to store the cookie. The Expression value field displays the following parameter value: ${rest.cookie['Set-Cookie: HSID=AQYQVn.DKrdst; Domain=.foo.com; Path=/; Expires=Wed, 13-Jan-2021 22:23:01 GMT; HttpOnly']}

For more information on HTTP cookie specifications and setting a HTTP cookie, see the online information sources.

**Note:** The field does not allow Java reserved words such as INT, CHAR, and BOOLEAN to be used as parameter names.

5. From the Name drop-down list in the Body type, select a parameter. This list displays the input and output parameters of the operation (ABL routine) that is associated with the REST resource verb.

**Note:** The Name and JSON type fields appear when you specify the node type as Body in the Type drop-down.

6. From the JSON type drop-down, select a data type. This list displays the OpenEdge supported JSON data types.

**Note:** The JSON type drop-down will be enabled only when you specify your own parameter instead of selecting one of the parameters listed in the drop-down.

You can see the added node entry under the selected HTTP request or response element. The defined node context menu provides the options to edit (Edit node) or delete (Delete) the node.

**See also**

Mapping REST resource verb parameters on page 217

---

**Deploying ABL REST Web applications**

**Starting and stopping Apache Tomcat**

The Progress Developer Studio for OpenEdge provides Apache Tomcat as a Java container to deploy your REST services and REST Web applications. When installing Progress Developer Studio for OpenEdge, Apache Tomcat is also installed and configured. The Tomcat will be located in your OpenEdge installation directory at $DLC/servers/tomcat.

Before you define and publish an OE Web Server or REST Service, you must start the Apache Tomcat. The REST adapter that expose the REST Web applications to the clients, run in the Java container.

To start the Apache Tomcat:
1. On the Windows Start > Programs menu, select Progress > OpenEdge > Proenv. A command shell window opens and displays information about your OpenEdge installation, followed by a command prompt:

```
proenv>
```

2. Enter the following command:

```
protc start
```

The Tomcat shell window opens.

**Note:** To stop the Apache Tomcat, enter `protc stop` at the `proenv>` command prompt.

Alternatively, you can start and stop the Apache Tomcat from the Servers view or from the OpenEdge Management and OpenEdge Explorer (OpenEdge Management and OpenEdge Explorer: Configuration guide).

**See also**
- Deploying a REST Web application on page 219
- Developing a REST Web application on page 210

**Creating an OE Web Server instance**

Use the New Server wizard to create a new an OE Web Server instance and define a connection to OpenEdge Explorer.

To create an OE Web Server instance:

1. From the Servers view in the OpenEdge Server perspective, right-click and choose New>Server on the context menu. The Define a New Server dialog appears, allowing you to choose OpenEdge AppServer, OE Web Server, or OpenEdge WebSpeed as the type of server you want to define.

2. To choose OE Web Server as the server type, either:
   - Type OE Web Server in the Select the server type field, which filters the results to show only available OE Web Servers.
   - Select Progress Software Corporation>OE Web Server as the server type from the list provided.

   The server host name localhost and Server name details are listed in the dialog.

3. Click Next.

4. Do one of the following:
   - Select the name of the connection from the OpenEdge Explorer connection drop-down list.
   - Click Configure. The Progress OpenEdge>Server>OpenEdge Explorer Connections Preferences page opens. For information about how to configure an OpenEdge Explorer connection, see Adding an OpenEdge Explorer connection.

5. Select the OE Web Server from the OE Web Server drop-down list, or click Refresh to list all the OE Web Servers.

   The list of OE Web Servers includes all OE Web Servers in alphabetical order, which are managed by the OpenEdge Explorer connection you specify.
Note: You cannot create a new OE Web Server in Progress Developer Studio for OpenEdge; to create an OE Web Server instance, use OpenEdge Explorer.

When you select the OE Web Server, the name of the server is updated in the **Server name** field.

6. Click **Next**. The **Add and Remove** dialog appears, showing all available REST Web applications from the workspace. See **Add and Remove** dialog on page 233.

7. To configure an available REST Web application, either select the application in the **Available** list and click **Add**, or double-click the application in the **Available** list. To configure all the available REST Web applications, click **Add All**.

   You can also remove a REST Web application from the **Configured** list by either selecting the application and clicking **Remove** or double-clicking the application in the **Configured** list. You can also select more than one REST Web application and click **Remove All**.

8. Click **Finish**.

   The defined OE Web Server instance appears in the **Servers** view.

**See also**

- Deploying REST services on page 219
- OE Web Server statistics on page 234

**Publishing a REST service**

You can publish a REST service using the Servers view in your OpenEdge Server perspective. To publish a REST service, create a server instance, associate the REST service to the defined server, and then publish the REST service to the Apache Tomcat Java container. See Associating REST services with an OE Web Server on page 224.

**Note:** After you republish a REST service, the service is enabled by default and becomes available for client requests.

To publish a REST service:

- From the **Servers** view, do one of the following:
  - Right-click a server instance, and then select **Publish** on the context menu.
  - Select a server instance, and then click **Publish to the server** icon on the **Servers** view toolbar.

This publishes the REST service associated with the selected OE Web Server to the Apache Tomcat Java container. For the published REST Service, a REST Web application WAR file is generated and deployed to the Java container. For example, if you use the Apache tomcat to deploy a REST service, a WAR file with the REST service name is created under the OpenEdge installation directory, `$DLC/servers/tomcat/webapps`. A separate WAR file is created for each deployed REST service.

For more information about the REST Web application WAR file layout, see the Progress OpenEdge Application Server: Administration guide.

You can disable or enable a deployed REST service from OpenEdge Management and OpenEdge Explorer. For more information about enabling or disabling a REST Web application, see the Progress OpenEdge Management and OpenEdge Explorer: Configuration guide.

You can view the status of the server in the Servers view beside the published REST service. For more information, see Status of OE Web Server and REST service on page 239.
After deploying the REST service, if you modify the artifacts of a project related to the deployed service, the status of the respective artifact’s module changes to Republish. You must republish the server manually if you have not set it to republish automatically so that the deployed service reflects the changes.

See also
Exposing an ABL application as a REST Web application on page 210
Starting the Tomcat Web server on page 219
Generating WAR file on page 222

Viewing REST service URI
When you deploy a REST service to the OE Web Server, the Progress OpenEdge Server Monitor view displays the Uniform Resource Identifier (URI) of the REST service.

To view the REST service URI:

1. Access the Servers view by selecting Window > Show View > Other > Server > Progress OpenEdge Server Monitor.
2. Place the cursor on the REST service whose URI you want to view.

The URI for the REST service appears in a tooltip. For more information, see the Understanding REST Web application URI construction section in the OpenEdge Application Server - Administration guide.

See also
Publishing a REST service on page 221

Packaging ABL REST Web applications

Exporting a REST Web application
The Export REST Application wizard allows you to package a REST service as a REST Web application (WAR file) or to export services incrementally by creating a .PAAR file for each REST service in the project. You can package one or more services into a single REST Web application. You can use WAR file to deploy the packaged REST Web application to the Apache Tomcat Java container.

To export as a WAR file or incremental service:

1. In the Project Explorer view, select a project that contains the REST services you want to package as a REST Web application.
2. Right-click the project, and then select Export > REST Application on the context menu.

Note: If you do not see the REST Application option on the right-click context menu, then select Export. On the Export wizard, select Progress OpenEdge > REST Application, and then click Next.

The Export REST Application wizard appears.

3. The REST project drop-down displays the project you have selected in step 1 on page 222. You can select a different project from the drop-down.

Note: The REST project drop-down lists only the projects of your current workspace.
4. In the **Destination** drop-down, specify a location or click **Browse** to select a different location for the REST Web application WAR or .PAAR file.

**Note:** The **Destination** drop-down displays a list of previous WAR file generation locations of your current workspace.

5. Select **OE Web Server** in the **Server type** field to specify the server to export the REST services.

6. Select one of the following options:

   - **Export as a WAR file** to export services by creating a single .WAR file for all the selected REST services in the project.
   - **Export services incrementally** to export services by creating a .PAAR file for each selected REST service in the project.

7. From the **REST and Data Object services** list, select the services that you want to export.

8. Click **Finish**.

The selected services are exported as a single REST Web application (WAR file) or as .PAAR files based on your selection and saved at the location specified in the **Destination** field. For information on the REST Web application WAR file layout, see the *OpenEdge Application Server - Administration* guide.

**See also**

- Building a REST Web application on page 208

### Working with OE Web Server

#### Defining an OE Web Server runtime environment

The **Server Runtime Environments** preferences page allows you to define an OE Web Server runtime environment.

To define an OE Web Server runtime environment:

1. Click **Window > Preferences**. The **Preferences** page appears.

2. On the left pane of the **Preferences** page, expand **Server > Runtime Environments**. The **Server Runtime Environments** page appears on the right pane.

3. Click **Add**. The **New Server Runtime Environment** dialog appears.

4. From the server runtime environment list, click **Progress Software Corporation > OpenEdge OE Web Server**.

5. Click **Finish**.

This creates the OE Web Server runtime environment. The OE Web Server runtime environment appears on the **Server Runtime Environments** preferences page.

**See also**

- Defining a OE Web Server on page 220
Associating REST services with an OE Web Server

The Add and Remove dialog allows you to associate or disassociate REST services with an OE Web Server instance.

By default, the OE Web Servers configured with the OpenEdge Explorer (OEE) or OpenEdge Manager (OEM) on the OpenEdge Explorer Connections page appear in the Servers view. The Add and Remove option available on the OE Web Server context menu allows you to associate or disassociate REST services with an OE Web Server instance.

Note: You must associate REST services with an OE Web Server instance before you can publish the REST services to the Apache Tomcat.

To associate REST services with an OE Web Server instance:

1. From the Servers view, select an OE Web Server instance with which you want to associate the REST services.
2. Select Add and Remove on the OE Web Server context menu.
   The Add and Remove dialog appears and displays a list of available REST services and another list of associated REST services.
3. Select the REST services from the Available list, and then click Add. This moves the selected REST services to the Configured list.
   Note: If you want to associate all the available REST services with an OE Web Server instance, click Add All. To remove an associated REST service, select it in the Configured list and click Remove. To remove all the associated REST services, click Remove All.
4. Select the If server if started, publish changes immediately check box to publish changes immediately upon the startup of the OE Web Server.
5. Click Finish to save your changes.

See also
Defining a OE Web Server on page 225
Publishing REST service on page 221
Add and Remove dialog on page 233

Viewing and modifying the OE Web Server instance properties

Using the OE Web Server Editor, you can view and modify the settings of an OE Web Server instance.

To view or modify an OE Web Server instance settings:

1. From the Servers view in the OpenEdge Server perspective, open the OE Web Server Editor in either of the following ways:
   • Double-click an OE Web Server instance.
   • Select an OE Web Server instance, right-click, and then select Open.
2. View or modify the OE Web Server properties in the following categories:
General Information
Connection
Publishing
Timeouts
OE Web Server Options

3. Press **CTRL+S** to save your changes.

See also
OE Web Server Editor on page 231

Renaming an OE Web Server instance
You can use the **Rename** option from **Servers** view to modify the name of an OE Web Server instance.

To rename an OE Web Server instance:
1. From the **Servers** view, select an OE Web Server.
2. Right-click the OE Web Server, and then select **Rename** on the context menu.
   - The selected OE Web Server is highlighted for inline editing.
3. Enter a new name of the OE Web Server, and then press **ENTER**.

See also
Working with OE Web Server on page 223

Using the Clean option
Selecting **Clean** undeploys REST services associated with the OE Web Server instance, and then republishes the services associated with the OE Web Server instance to the Java container.

To use the clean option before republishing:
1. From the **Servers** view, right-click an OE Web Server instance.
2. Select **Clean** on the context menu.
3. Click **Yes** to confirm cleaning when prompted.

See also
Publishing REST service on page 221
Working with OE Web Server on page 223
Working with Servers view on page 226

Deleting an OE Web Server instance
You can use the **Delete** option from **Servers** view to delete an OE Web Server instance that you have defined.

To delete an OE Web Server instance:
1. From the **Servers** view, select an OE Web Server that you want to delete.
2. Right-click the OE Web Server, and then select **Delete** on the context menu.
3. Click **Yes** at the confirmation prompt to delete the selected OE Web Server.

This deletes the OE Web Server instance and its associated REST services from the **Servers** view.

**Note:** The instance of the OE Web Server and its associated REST services deployed to the Apache Tomcat Java container will not be deleted.

---

**See also**

*Working with OE Web Server* on page 223

---

## Working with REST services

### Undeploying REST services

The **Remove** context option allows you to undeploy a REST service from the Apache Tomcat Java container.

To undeploy a REST Service:

1. From the **Servers** view, select a REST Service that you want to undeploy.
2. Right-click a REST Service, and then select **Remove** on the context menu.
3. Click **OK** at the confirmation prompt.

This undeploys the selected REST service from the Apache Tomcat.

**See also**

*Creating REST Service* on page 211

*Publishing REST service* on page 221

---

### Deleting a REST service

You can use the **Delete** option to remove a defined REST service from your project.

To delete a REST service:

1. From the **Project Explorer** view, expand the **Defined Services** node under your REST project.
2. Right-click a REST service, and then select **Delete** on the context menu.
   
   A confirmation dialog prompts you to confirm the deletion.

3. Click **Yes** to confirm the deletion.

The selected REST service and its artifacts are deleted from the project. The selected project is automatically refreshed.

**See also**

*Defining a REST service* on page 211
Reference

New REST Service dialog

The New REST Service dialog allows you to create a REST service. Access this dialog by clicking New > REST Service on the File menu.

The New REST Service dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service name</td>
<td>Specify a name for the new REST service.</td>
</tr>
<tr>
<td>Service description</td>
<td>Specify a description for the REST service.</td>
</tr>
<tr>
<td>Service relative URI</td>
<td>Specify a relative path for all the resources to be defined in the REST</td>
</tr>
<tr>
<td></td>
<td>service. For example, /Customer.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The relative URL must start with &quot;/&quot;.*</td>
</tr>
<tr>
<td>Project</td>
<td>Select a project where you want all the REST service artifacts to be</td>
</tr>
<tr>
<td></td>
<td>generated. Click icon to select a different project from the Select Project</td>
</tr>
<tr>
<td></td>
<td>dialog.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> By default, the project name you select when creating the</td>
</tr>
<tr>
<td></td>
<td>REST service is displayed in the Project field.</td>
</tr>
<tr>
<td>OE Web Servers</td>
<td>Select the OE Web Server instances with which you want to associate</td>
</tr>
<tr>
<td></td>
<td>the defined REST service.</td>
</tr>
<tr>
<td>Finish</td>
<td>Creates a REST service along with the REST artifacts under the selected</td>
</tr>
<tr>
<td></td>
<td>OpenEdge project.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The REST service appears under the Defined Services node</td>
</tr>
<tr>
<td></td>
<td>in your OpenEdge project.</td>
</tr>
</tbody>
</table>

See also

Creating a REST service on page 211

REST Expose Editor on page 228
REST Expose Editor dialogs

REST Expose Editor

The REST Expose Editor allows you to define mapping between an REST-annotated ABL class and procedure file and a defined REST service.

Note: The REST Expose Editor opens after you have successfully defined a REST service. If you have closed the REST editor, you can re-open the editor by choosing Edit on the REST Service right-click context menu or double-clicking the REST Service. See Opening the REST Expose Editor on page 212.

The REST Expose Editor includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service relative URI</td>
<td>Displays the relative URI of the current REST Service open in the REST editor. You can modify the relative path by clicking Edit which opens the Edit Service Relative URI dialog.</td>
</tr>
<tr>
<td></td>
<td>Note: The relative URL must start with &quot;/&quot;&quot;, for example, /&lt;application relative path&gt;.</td>
</tr>
<tr>
<td>Add Resource</td>
<td>Opens the New REST Resource dialog which allows you to add a REST resource to the REST service currently open in the REST editor. Each REST resource you add is associated with these four REST (HTTP) verbs: Get, Put, Post, and Delete.</td>
</tr>
<tr>
<td>Delete Resource</td>
<td>Deletes a selected REST resource.</td>
</tr>
<tr>
<td></td>
<td>Note: When you delete a REST resource, all its child resources are deleted.</td>
</tr>
<tr>
<td>Resources</td>
<td>Displays a list of the REST resources added to the current REST Service.</td>
</tr>
<tr>
<td>Verb Association</td>
<td>Displays the REST verbs (HTTP methods) for a REST resource selected on the Resources list.</td>
</tr>
<tr>
<td></td>
<td>Opens the Associate Operation With Verb dialog which allows you to associate a REST verb with an operation (REST interface-annotated ABL class or procedure file).</td>
</tr>
</tbody>
</table>
### Option

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Delete" /></td>
<td>Deletes the operation associated with a REST verb.</td>
</tr>
</tbody>
</table>

### Mapping Definitions

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Input" /></td>
<td>Displays the parameter mapping section for a selected REST resource verb. This section contains the following two tabs:</td>
</tr>
<tr>
<td><img src="image" alt="Input" /></td>
<td>• <strong>Input</strong> – Allows you to map the REST input parameters to the input parameters of the associated ABL routine (interface parameters).</td>
</tr>
<tr>
<td><img src="image" alt="Input" /></td>
<td>• <strong>Output</strong> – Allows you to map the REST output parameters to output parameters of the associated ABL routine (interface parameters).</td>
</tr>
</tbody>
</table>

**Note:** You must associate a REST resource verb with a REST-annotated ABL class or procedure file, to populate the Mapping Definitions section with the interface parameters.

### See also

- Working with the REST Expose Editor on page 212
- Creating a REST Web application on page 211
- REST Expose Editor dialogs on page 228

### Edit Service Relative URI dialog

The **Edit Service Relative URI** dialog allows you to modify the relative path of a defined REST Service.

Access the **Edit Service Relative URI** dialog by clicking **Edit** next to the **Service relative URI** field on the **REST Expose Editor**.

The **Edit Service Relative URI** dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Service relative URI</strong></td>
<td>Specify a new relative path for the REST Service. This is the path for all the resources defined in your REST Service. The relative path must start with &quot;&quot;. For example, &quot;/&lt;service relative path&gt;&quot;.</td>
</tr>
<tr>
<td><strong>OK</strong></td>
<td>Displays the modified relative URL in the <strong>Service relative URI</strong> field on the <strong>REST Expose Editor</strong>.</td>
</tr>
</tbody>
</table>

### See also

- Editing service relative URI on page 213
- REST Expose Editor on page 228

### New REST Resource dialog

The **New REST Resource** dialog allows you to add REST resources to a defined REST Service.
Access this dialog by clicking Add Resource icon on the REST Expose Editor.

The New REST Resource dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource URI</td>
<td>Specify a resource URI for the REST resource you want to create. The REST resources you add are identified with Uniform Resource Identifier (URI). The resource URI must start with &quot;/&quot;. For example, &quot;/&lt;REST resource name&gt;&quot;.</td>
</tr>
<tr>
<td>Note:</td>
<td>In a REST resource URI, you cannot use the Java reserved words (such as int, char, or boolean) as the path or query parameter names. For more information, see Adding REST parameters on page 214.</td>
</tr>
<tr>
<td>OK</td>
<td>Adds the REST resource to the Resources list on the REST editor.</td>
</tr>
</tbody>
</table>

See also
Adding a REST resource on page 214
REST Expose Editor on page 228

Associate Operation With Verb dialog

The Associate Operation With Verb dialog allows you to associate an operation (REST interface-annotated ABL class or procedure file) with a verb of a REST resource.

Access this dialog by clicking the icon present beside a REST verb on the REST Expose Editor.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resources</td>
<td>Displays a list of REST interface-annotated ABL class (.cls) and procedures (.p) files.</td>
</tr>
<tr>
<td>Note:</td>
<td>All the REST interface-annotated ABL class and procedure files available in your current project are displayed under the Resources list.</td>
</tr>
<tr>
<td>Procedures/Functions</td>
<td>Displays a list of internal procedures and functions of the REST interface-annotated ABL class or procedure file selected on the Resources list.</td>
</tr>
<tr>
<td>OK</td>
<td>Associates the operation to the selected verb of a REST resource. The selected operation appears beside the associated verb in the Verb Association section.</td>
</tr>
</tbody>
</table>

See also
Mapping REST methods on page 216
OE Web Server related dialogs and context menu

OE Web Server Editor

You use the **OE Web Server Editor** to view or modify an OE Web Server instance properties that define the connection to the OpenEdge Explorer and the REST broker.

From the **Servers** view in the OpenEdge Server perspective, you can access the **OE Web Server Editor** in either of the following ways:

- Double-click an OE Web Server instance.
- Select an OE Web Server instance, right-click, and then select **Open**.

The **OE Web Server Editor** provides information in the following categories:

- **General Information** - Provides the host name and other common settings.
- **Connection** - Specifies the information for connection to the OpenEdge Explorer.
- **Publishing** - Specifies the publish settings.
- **Timeouts** - Specifies the time limit to complete server operations (Start and Stop).

General Information

You can modify the following General Information properties in the **OE Web Server Editor**:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Server name</strong></td>
<td>The name of the OE Web Server.</td>
</tr>
<tr>
<td><strong>Host name</strong></td>
<td>The name of the host machine.</td>
</tr>
<tr>
<td><strong>Runtime Environment</strong></td>
<td>The current runtime environment. You can click the drop-down list or the Runtime Environment link to change to a different OpenEdge AVM runtime or OpenEdge version.</td>
</tr>
</tbody>
</table>

Connection

You can modify the following OpenEdge Explorer properties in the **OE Web Server Editor**:
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OpenEdge Explorer connection</strong></td>
<td>The name of an existing OpenEdge Explorer connection. To add a connection, click <strong>Configure</strong>.</td>
</tr>
<tr>
<td>Broker</td>
<td>To choose from a list of available REST brokers, click the drop-down arrow. To refresh the list, click Refresh.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>Remember that you cannot create a new REST broker in Progress Developer Studio for OpenEdge; to create a broker, use OpenEdge Management.</td>
</tr>
</tbody>
</table>

**Publishing**

You can modify the publishing settings to choose one of the following options:

- Never publish automatically
- Automatically publish when resources change
- Automatically publish after a build event

You can also set the publishing interval in seconds.

**Timeouts**

You can specify the time limit (in seconds) for completing Start and Stop server operations.

**Note:** You can start and stop the OE Web Server from the **Servers** view using the **Start** and **Stop** options on the OE Web Server's context menu.

**See also**

*Viewing and modifying the OE Web Server instance properties* on page 224

**OE Web Server context menu**

Right-clicking an OE Web Server instance in the **Servers** view displays the OE Web Server context menu. This menu has the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>New</td>
<td>Creates a new OE Web Server instance.</td>
</tr>
<tr>
<td>Open</td>
<td>Opens the <strong>OE Web Server Editor</strong> on page 231.</td>
</tr>
<tr>
<td>Copy</td>
<td>Copies the selected OE Web Server.</td>
</tr>
<tr>
<td>Delete</td>
<td>Deletes an OE Web Server instance.</td>
</tr>
<tr>
<td>Rename</td>
<td>Modifies the name of a selected OE Web Server.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Restart</td>
<td>Restarts the selected OE Web Server instance.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This option will be available only when an OE Web Server instance is started and running.</td>
</tr>
<tr>
<td>Start/Stop</td>
<td>Starts or stops the selected OE Web Server instance.</td>
</tr>
<tr>
<td>Publish</td>
<td>Publishes the selected OE Web Server to the Apache Tomcat Java container. See Publishing a REST service on page 221.</td>
</tr>
<tr>
<td>Clean</td>
<td>Removes any invalid resources from the server before doing a republish. See Using the Clean option on page 225.</td>
</tr>
<tr>
<td>Add and Remove</td>
<td>Opens the Add and Remove dialog on page 233 which allows you to associate the REST services with a selected OE Web Server instance.</td>
</tr>
<tr>
<td>Monitoring</td>
<td>Opens the Monitoring properties page which you can use to monitor server ports to view data sent and received over the ports of the selected OE Web Server.</td>
</tr>
<tr>
<td>OpenEdge Explorer</td>
<td>Opens the OpenEdge Explorer page. When started, OpenEdge Explorer prompts for an administrator user name and password. The default is admin for both. After login, it runs in a browser embedded in the active perspective.</td>
</tr>
<tr>
<td>Server Monitor</td>
<td>Opens the Progress OpenEdge Server Monitor view. This view allows you to view the status and properties of the deployed OE Web Server instance and REST services.</td>
</tr>
<tr>
<td>Properties</td>
<td>Opens the General properties page for the selected OE Web Server. This page contains general information of the OE Web Server such as Name, Type, Vendor, and Location.</td>
</tr>
</tbody>
</table>

### Add and Remove dialog

The **Add and Remove** page of the **New Server** wizard allows you to associate or disassociate REST services with an OE Web Server instance.

This page includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available</td>
<td>Displays a list of available REST services.</td>
</tr>
<tr>
<td>Configured</td>
<td>Displays a list of configured REST services.</td>
</tr>
<tr>
<td>Add</td>
<td>Moves a selected REST service to the configured list.</td>
</tr>
<tr>
<td>Add All</td>
<td>Moves all available REST services to the configured list.</td>
</tr>
<tr>
<td>Remove</td>
<td>Moves a selected REST service to the available list.</td>
</tr>
</tbody>
</table>
## Progress OpenEdge Server Monitor view

The **Progress OpenEdge Server Monitor** view provides the properties and statistics of the OE Web Server instances and its associated REST services, defined in the **Servers** view. It is the default view that appears in the console area of the OpenEdge Server perspective.

If you do not see the **OpenEdge Progress Server Monitor** view, you can open the view from the main menu by selecting `Window > Show View > Progress OpenEdge Server Monitor`.

### OE Web Server statistics

The **Progress OpenEdge Server Monitor** view provides summary of a selected OE Web Server instance on the **Summary** tab and statistics on the **Statistics** tab.

**Note:** The properties and statistics displayed in this view are read-only. You can modify these properties from the OpenEdge Management console. See *OpenEdge Management and OpenEdge Explorer: Configuration* guide.

### Summary tab

This tab displays the following runtime properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of a selected OE Web Server instance.</td>
</tr>
<tr>
<td>Description</td>
<td>The description of an OE Web Server instance.</td>
</tr>
<tr>
<td>Admin Enabled</td>
<td>Indicates whether administrator privileges are in effect, either True or False.</td>
</tr>
<tr>
<td>WebApp Enabled</td>
<td>Controls the OE Web Server's ability to accept and process requests to any of its deployed OE Web Server services, either True or False.</td>
</tr>
<tr>
<td>Log Append</td>
<td>Indicates whether to start a new log each time an OE Web Server is restarted, either True or False.</td>
</tr>
<tr>
<td>Log File</td>
<td>The location and name of the OE Web Server log file.</td>
</tr>
</tbody>
</table>

---

### Option Table

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove All</td>
<td>Moves all the associated REST services to the available list.</td>
</tr>
<tr>
<td>Finish</td>
<td>Finish adding and removing REST services.</td>
</tr>
</tbody>
</table>

**See also**

- [Associating REST services with a OE Web Server](#) on page 224
- [Defining OE Web Server](#) on page 220
## Logging Level

The level of information to be written to the OE Web Server log. The logging levels are:

- **0 - None.** Log no entries. This is equivalent to turning logging off.
- **1 - Errors.** Log OpenEdge error messages. This includes all error messages and is unrelated to the entry types specified. Errors continue to be logged at all higher levels.
- **2 - Basic.** Logging Entry Types determine the logged information. Each entry type generates at least some output. This is the default.
- **3 - Verbose.** Logging entry types determine the logged information, but it is typically more information than Basic.
- **4 - Extended.** Logging entry types determine the logged information, but it is typically more information than Verbose.

## Manager Properties URI

The Uniform Resource Identifier (URI) for the properties of an OE Web Server instance.

## Properties File Name

The location and name of the properties `oerm.properties` file.

## Statistics tab

This tab displays the following runtime statistics:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active requests</td>
<td>Requests currently being serviced by an OE Web Server.</td>
</tr>
<tr>
<td>Authentication errors</td>
<td>The total number of authentication errors.</td>
</tr>
<tr>
<td>HTTP requests</td>
<td>The total requests received from the HTTP listener, including administrative and REST requests.</td>
</tr>
<tr>
<td>HTTP request errors</td>
<td>The total HTTP request errors returned by an OE Web Server.</td>
</tr>
<tr>
<td>Method not allowed errors</td>
<td>Total errors returned by the OE Web Server, with error counts broken out at the bottom of the list for each of several error categories when total errors are greater than zero (0).</td>
</tr>
<tr>
<td>REST disabled errors</td>
<td>The total number of errors while OE Web Server is disabled.</td>
</tr>
<tr>
<td>Services disabled</td>
<td>The total number of errors while REST service id disabled.</td>
</tr>
<tr>
<td>URL not found errors</td>
<td>The total number requests to unidentified resources.</td>
</tr>
<tr>
<td>Creation time</td>
<td>The date and time that counting began for the statistics.</td>
</tr>
<tr>
<td>Start time</td>
<td>The start date and time of an OE Web Server instance.</td>
</tr>
</tbody>
</table>
See also
Progress OpenEdge Server Monitor view on page 234

REST service statistics
The Progress OpenEdge Server Monitor view provides summary of a selected REST service on the Summary tab, statistics on the Statistics tab, and properties on the Properties tab. The REST service properties are stored in the <service name>.props file.

Note: The summary, statistics, and properties displayed in this view are read-only. You can modify these properties from the OpenEdge Management console. See OpenEdge Management and OpenEdge Explorer: Configuration guide.

Summary tab
This tab displays the following summary details:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of a selected REST service.</td>
</tr>
<tr>
<td>REST service URL</td>
<td>The REST service Uniform Resource Identifier (URI).</td>
</tr>
<tr>
<td>Enabled</td>
<td>Indicates if the application is enabled or disabled for client access.</td>
</tr>
</tbody>
</table>

Properties tab
This tab displays the following runtime properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppService Host</td>
<td>The host name for the NameServer or AppServer that supports a REST service. Defaults to localhost.</td>
</tr>
<tr>
<td>AppService Name</td>
<td>The name of an application service supported by the specified NameServer or AppServer and that supports all of the operations that define the service. Defaults to asbroker1.</td>
</tr>
<tr>
<td>AppService Port</td>
<td>The port number to access the REST service supported by the host specified by the appServiceHost property. Defaults to 5162.</td>
</tr>
<tr>
<td>AppService Protocol</td>
<td>The protocol that the REST Adapters use to access the host specified by the appServiceHost property. For example, for a NameServer host, this is AppServer; for an AppServer host, AppServerDC. Defaults to AppServer.</td>
</tr>
<tr>
<td>Connection Lifetime</td>
<td>The maximum lifetime (in seconds) of AppServer connections in the connection pool for this service.</td>
</tr>
<tr>
<td>Idle Session Timeout</td>
<td>The duration (in seconds) between attempts by the adapter to shut down extra network connections to the AppServer, based on client demand.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Initial Sessions</td>
<td>The number of network sessions to be created (and shared by all clients) when the connection pool for the service is initialized by the adapter.</td>
</tr>
<tr>
<td>Max Sessions</td>
<td>The maximum number of connected sessions allowed in the service connection pool.</td>
</tr>
<tr>
<td>Min Idle Connections</td>
<td>The minimum number of idle, or inactive, AppServer connections to maintain.</td>
</tr>
<tr>
<td>Min Sessions</td>
<td>The minimum number of connected sessions that the adapter attempts to maintain in the service connection pool.</td>
</tr>
<tr>
<td>No Host Verify</td>
<td>Controls whether the service verifies that the host name of the connecting AppServer matches the Common Name specified in the server digital certificate (meaningful only in the context of SSL connections).</td>
</tr>
<tr>
<td>No Session Reuse</td>
<td>Controls whether the service requests reuse of the SSL session ID when reconnecting to the same AppServer (meaningful only in the context of SSL connections).</td>
</tr>
<tr>
<td>NS Client Max Port</td>
<td>The maximum value for the adapter to specify for the port number used to communicate with a NameServer.</td>
</tr>
<tr>
<td>NS Client Min Port</td>
<td>The minimum value for the adapter to specify for the port number used to communicate with a NameServer randomly. This property applies only to services that use a NameServer to access application services on an AppServer.</td>
</tr>
<tr>
<td>NS Client Picklist Expiration</td>
<td>The maximum duration (in seconds) that the adapter retains a list of AppServer options (pick list) for an idle application service.</td>
</tr>
<tr>
<td>NS Client Picklist Size</td>
<td>The maximum duration (in seconds) that the adapter retains a list of AppServer options (pick list) for an idle application service.</td>
</tr>
<tr>
<td>NS Client Port Retry</td>
<td>The maximum number of requests that the adapter makes for a valid local port number when attempting to communicate with a NameServer.</td>
</tr>
<tr>
<td>NS Client Port Retry Interval</td>
<td>The interval (in milliseconds) that the adapter waits between requests to get a valid port number when attempting to communicate with a NameServer.</td>
</tr>
<tr>
<td>Request Wait Timeout</td>
<td>Determines how the adapter handles requests when the service connection pool becomes full.</td>
</tr>
<tr>
<td>Service Logging Level</td>
<td>The amount and type of information written to the service log each log entry. Defaults to 2.</td>
</tr>
<tr>
<td>Stale O4GL Object Timeout</td>
<td>The maximum duration (in seconds) that a service object can be idle before it is released.</td>
</tr>
<tr>
<td>Wait If Busy</td>
<td>Determines how to handle client requests to a service that is busy processing a prior request.</td>
</tr>
</tbody>
</table>
Statistics tab
This tab displays the following statistics:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Successful User Requests</td>
<td>The total number of successful user requests handled by the selected REST service.</td>
</tr>
<tr>
<td>Number of Failed User Requests</td>
<td>The total number of requests that fail because of missing request information.</td>
</tr>
<tr>
<td>Number of Successful Admin Requests</td>
<td>The total number of successful administrator requests.</td>
</tr>
<tr>
<td>Number of Failed Admin Requests</td>
<td>The total number of administrator requests that fail because of missing request information.</td>
</tr>
<tr>
<td>Average Response Time for User Requests</td>
<td>The average response time of each request response.</td>
</tr>
<tr>
<td>Maximum Response Time for User Requests</td>
<td>The maximum response time of each request response.</td>
</tr>
<tr>
<td>Number of Successful User Logins</td>
<td>The total number of successful user logins.</td>
</tr>
<tr>
<td>Number of Failed User Logins</td>
<td>The total number of failed user logins.</td>
</tr>
<tr>
<td>Number of Successful Admin Logins</td>
<td>The total number of successful administrator logins.</td>
</tr>
<tr>
<td>Number of Failed Admin Logins</td>
<td>The total number of failed administrator logins.</td>
</tr>
<tr>
<td>Number of Successful AppServer Connections</td>
<td>The total number of successful AppServer connections in the connection pool.</td>
</tr>
<tr>
<td>Number of Failed AppServer Connections</td>
<td>The total number of failed AppServer connections in the connection pool.</td>
</tr>
<tr>
<td>Number of Adapter Errors</td>
<td>The total number of REST adapter failures.</td>
</tr>
<tr>
<td>Number of Application Errors</td>
<td>The total number of errors related to REST services.</td>
</tr>
<tr>
<td>StartTime</td>
<td>The start date and time of the REST service.</td>
</tr>
<tr>
<td>Creation Time</td>
<td>The date and time that counting began for the statistics.</td>
</tr>
</tbody>
</table>

See also
Progress OpenEdge Server Monitor view on page 234
Status of OE Web Server and REST service

You can use the Servers view to view the current status of an OE Web Server instance and its associated REST services.

The Servers view allows you to manage the OE Web Server instances. This view displays a list of all your OE Web Server instances and REST services that are associated with that OE Web Server instance.

Access the Servers view by selecting Window > Show View > Servers from the main menu.

The following table lists the possible statuses:
### Status and Description

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Stopped]</td>
<td>The OE Web Server instance is stopped and not connected to the Apache Tomcat Java container. You cannot deploy REST service and REST Web applications to the Apache Tomcat Java container.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Select <strong>Start</strong> on an OE Web Server instance context menu to start the OE Web Server instance.</td>
</tr>
<tr>
<td>[Starting]</td>
<td>The OE Web Server instance is in the process of connecting to the Apache Tomcat and is beginning the server process.</td>
</tr>
<tr>
<td>[Started]</td>
<td>The OE Web Server instance is successfully connected to the Apache Tomcat. Both the OE Web Server instance and Apache Tomcat are ready to deploy services.</td>
</tr>
<tr>
<td>[Started, Synchronized]</td>
<td>Both the OE Web Server instance and its associated REST services are in sync. REST services are copied to a location on the Web applications folder. For example, in case of Apache Tomcat, the REST services are published to the Web applications folder in your OpenEdge installation directory, <code>$DLC/servers/tomcat/webapps</code>. A separate WAR file is created for each deployed REST service.</td>
</tr>
<tr>
<td>[Stopped, Synchronized]</td>
<td>The OE Web Server instance is in stop state. Both the OE Web Server instance and its associated REST services are in sync.</td>
</tr>
<tr>
<td>[Started, Republish]</td>
<td>Either the server or the associated services, or both have changed. When the server is restarted, the modified services are also republished.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The status appears as Republish beside the modified REST service. Select <strong>Republish</strong> on the OE Web Server instance context menu to republish the REST service.</td>
</tr>
<tr>
<td>[Synchronized] [Not Published]</td>
<td>Both the OE Web Server instance and its associated REST services are in sync. The REST service is not deployed to the Apache Tomcat Java container and disabled for client access.</td>
</tr>
<tr>
<td>[Synchronized] [Published]</td>
<td>Both the OE Web Server instance and its associated REST services are in sync. The REST service is deployed to the Apache Tomcat Java container and enabled for client access.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> When a REST service is deployed to a Apache Tomcat Java container, the deployed REST service is enabled by default and will be available for client access.</td>
</tr>
</tbody>
</table>
### Status and Description

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Synchronized] [OE Web Server is not started]</td>
<td>Both the OE Web Server instance and its associated REST services are in sync. But, the OE Web Server instance is not started. Right-click the OE Web Server instance, and then select <strong>Start</strong> on the context menu to start the OE Web Server.</td>
</tr>
<tr>
<td>[Republish]</td>
<td>Indicates that a REST service is modified and must be re-deployed to the Apache Tomcat Java container.</td>
</tr>
<tr>
<td>[Not Published]</td>
<td>Indicates that the OE Web Server instance is started but the <strong>Publish changes immediately</strong> check box was not selected on the <strong>Define AppServer content module</strong> page while creating the REST project. Select <strong>Publish</strong> on the OE Web Server context menu to publish the REST services having the status as Not Published.</td>
</tr>
</tbody>
</table>

**Note:** Before you publish or republish the REST services to the Apache Tomcat Java container, ensure that the selected OE Web Server instance is started. You can start or stop the OE Web Server instance from the **Servers** view using the **Start** and **Stop** option on the OE Web Server instance context menu or from the **Servers** view toolbar.

### See also
- OpenEdge Server Monitor view on page 234
- Publishing REST services on page 221
- OE Web Server context menu on page 232

### OE Web Server and REST service logs

You can display an OE Web Server instance log file and a REST service log file from the **Console** view. By default, the **Console** view is available in the OpenEdge Server perspective for a REST project.

**Note:** Ensure that the OE Web Server instance is started and REST service is enabled to view their logs. By default, the REST service is enabled when it is published to the Apache Tomcat Java container. For more information on enabling or disabling a REST service/application, see the OpenEdge Management and OpenEdge Explorer: Configuration guide.

You can open the **Console** view to display an OE Web Server log file and a REST service log file in any of the following ways:

- From the **Servers** view, right-click an OE Web Server instance. Then, choose **Show In > Console**.
- From the main menu, select **Window > Show View > Console** view, click **Display Selected Console** from the toolbar.
- From the main menu, select **Window > Show View > Console** and from the **Console** view, click **Display Selected Console** from the toolbar and select an OE Web Server instance log or a REST service log from the drop-down list.

**See also**
- Status of OE Web Server and REST service on page 239
Overview of ABLUnit testing framework

ABLUnit is a unit testing framework for the Advanced Business Language (ABL) programs. It is similar to JUnit-based and XUnit-based unit testing frameworks.

ABLUnit for Progress Developer Studio for Open Edge helps you to write and run repeatable unit test cases or test suites. The automated testing of your programs checks if the code is working as intended and is valid. It helps you to:

- Identify bugs in your code.
- Write modular and loosely-coupled code.
- Test the code in parts

The ABLUnit testing framework supports two types of test cases:

- Test Class and Test Suite Class (.cls file)
- Test Procedure and Test Suite Procedure (.p file)

ABLUnit testing framework includes the following components:

- Annotations for marking the test cases
- Assertions for testing expected results
- Test suites for easily organizing and running test cases
- Test runners for running a test case or test suite from the command line or launch configuration
Results view for viewing and analyzing the test results

For details, see the following topics:

- Concepts
- Tasks
- Reference

**Concepts**

**Overview of ABLUnit testing framework**

ABLUnit is a unit testing framework for the Advanced Business Language (ABL) programs. It is similar to JUnit-based and XUnit-based unit testing frameworks.

ABLUnit for Progress Developer Studio for Open Edge helps you to write and run repeatable unit test cases or test suites. The automated testing of your programs checks if the code is working as intended and is valid.

It helps you to:

- Identify bugs in your code.
- Write modular and loosely-coupled code.
- Test the code in parts

The ABLUnit testing framework supports two types of test cases:

- Test Class and Test Suite Class (.cls file)
- Test Procedure and Test Suite Procedure (.p file)

ABLUnit testing framework includes the following components:

- Annotations for marking the test cases
- Assertions for testing expected results
- Test suites for easily organizing and running test cases
- Test runners for running a test case or test suite from the command line or launch configuration
- Results view for viewing and analyzing the test results

**Test Class**

A test class is an ABL class that contains one or more test methods. All the naming requirements of the test class are similar to ABL class.
Here is an example of an ABL class file; any method in a test class can be converted to a test method by adding @Test annotation as shown in the example.

```abl
USING Progress.Lang.*.
CLASS MyClass:
  METHOD PUBLIC CHARACTER M1( ):
    DEFINE VARIABLE result AS CHARACTER NO-UNDO.
    result = "M1".
    RETURN result.
  END METHOD.
  METHOD PUBLIC INTEGER M2( ):
    DEFINE VARIABLE result AS INTEGER NO-UNDO.
    result = 10.
    RETURN result.
  END METHOD.
END CLASS.
```

Here is the ABLUnit test class for the above class file:

```abl
ROUTINE-LEVEL ON ERROR UNDO, THROW.
USING OpenEdge.Core.Assert.
CLASS MyTestClass:
  @Test.
  METHOD PUBLIC VOID TestM1( ):
    DEFINE VARIABLE obj AS MyClass NO-UNDO.
    DEFINE VARIABLE returned AS CHARACTER NO-UNDO.
    obj = new MyClass().
    returned = obj:M1().
    Assert:equals("M1", returned).
  END METHOD.
  @Test.
  METHOD PUBLIC VOID TestM2( ):
    DEFINE VARIABLE obj AS MyClass NO-UNDO.
    DEFINE VARIABLE i AS INTEGER NO-UNDO.
    obj = new MyClass().
    i = obj:M2().
    Assert:equals(0,i).
  END METHOD.
END CLASS.
```

### Test Suite Class

A test suite class is an empty ABL class. It contains a list of test classes, test suite classes, test procedures, and test suite procedures. It helps you to group and organize test classes. When a test suite class is run, all the test classes, test procedures, test suite classes, and test suite procedures added in the test suite class will be run.

The ABL class is identified as a test suite by the @TestSuite annotation and you can add the test classes to this test suite as comma separated values to the classes attribute and test procedures as comma separated values to the procedures attribute as shown in the examples below:

```abl
@TestSuite(classes= "Org.test.TestClass1, Org.test.TestClass2").
@TestSuite(procedures= "Org/test/test_proc1.p, Org/test/test_proc2.p").
@TestSuite(classes= "Org.test.TestClass2, Org.test.TestClass4", procedures= "Org/test/test_proc1.p, Org/test/test_proc2.p").
```

If @TestSuite annotation is present, only the classes and procedures attribute values are read for identifying the test cases and the business logic is not executed.
Here is an example of test suite:

```
USING Progress.Lang.*.
ROUTINE-LEVEL ON ERROR UNDO, THROW.
@TestSuite(procedures="test.p").
@TestSuite(procedures="testLogin.p").
@TestSuite(classes="testcls.cls").
CLASS testSuite:
  END CLASS..
```

**Test Procedure**

A test procedure is an ABL procedure file that has one or more internal test procedures. All the naming requirements of the test procedure are similar to ABL procedures.

Here is an example of an ABL procedure file; any procedure can be converted to a test procedure by adding @Test annotation as shown in the example.

```
PROCEDURE P1:
  DEFINE OUTPUT PARAMETER out AS INTEGER.
  out = 20.
END PROCEDURE.
```

The following is the test procedure for the above procedure.

```
ROUTINE-LEVEL ON ERROR UNDO, THROW.
USING OpenEdge.Core.Assert.
@Test.
PROCEDURE TestP1:
  DEFINE VARIABLE out AS INTEGER.
  RUN p1 (OUTPUT out).
  Assert:equals(20,out).
END PROCEDURE.
```

**Test Suite Procedure**

A test suite procedure is an empty ABL procedure. It contains a list of test classes, test suite classes, test procedures, and test suite procedures. When you run a test suite, all the test procedures and test suite are also run by default.

The test suite procedure is identified as a test suite by the @TestSuite annotation and you can add the test procedures to this test suite as comma separated values to the classes or procedures attribute, similar to test suite class.

If @TestSuite annotation is present, only the classes and procedures attribute values are read for identifying the test cases to run and the business logic, if any, in the procedure is ignored.

Here is an example of test suite procedure:

```
@TestSuite(procedures="TestProcedure.p").
```

**Annotations supported with ABLUnit**

The ABLUnit testing framework uses annotations to identify and execute the test methods or internal procedures in a test case.
The following table gives an overview of the available annotations in the ABLUnit testing framework:

<table>
<thead>
<tr>
<th>Component</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Test</td>
<td>Identifies that a method or a procedure is a test method or procedure.</td>
</tr>
<tr>
<td>@Setup</td>
<td>Executes the method or procedure before each test. This annotation prepares the test environment such as reading input data or initializing the class.</td>
</tr>
<tr>
<td>@TearDown</td>
<td>Executes the method or procedure after each test. This annotation cleans up the test environment such as deleting temporary data or restoring defaults.</td>
</tr>
<tr>
<td>@Before</td>
<td>Executes the method or procedure once per class, before the start of all tests. This annotation can be used to perform time-sensitive activities such as connecting to a database.</td>
</tr>
<tr>
<td>@After</td>
<td>Executes the method or procedure once, after all the tests are executed. This annotation is used to perform clean-up activities such as disconnecting from a database.</td>
</tr>
<tr>
<td>@Ignore</td>
<td>Ignores the test. You can use this annotation when you are still working on a code, the test case is not ready to run, or if the execution time of test is too long to be included.</td>
</tr>
<tr>
<td>@Test (expected=&quot;ExceptionType&quot;)</td>
<td>Fails the method if the method does not throw the exception mentioned in the expected attribute.</td>
</tr>
</tbody>
</table>

**Lifecycle of ABLUnit framework**

The ABLUnit testing framework uses annotations to specify that a method in a test class or an internal procedure in a test procedure is a test case.

The @Test annotation is used to mark a method or internal procedure as a test case. The testing framework identifies the test case using the @Test annotation and executes the test case.

The scope of a test case is defined inside a test class or test procedure. A test class or procedure is annotated using @Before and @After.

The ABLUnit testing framework can be invoked, for example, by providing a test class or a test procedure. The framework identifies the test cases by looking for test annotations associated with methods or procedures. The execution of test cases proceeds as defined below:

1. If a method or procedure marked with @Before annotation is present, it is executed before any test cases.
2. If a method or procedure marked with @After annotation is present, it is executed after executing all the test cases.
3. For each test method or procedure:
   - @Setup annotated method or procedure is executed, if there is any.
   - The test method or procedure is executed.
   - @TearDown annotated method or procedure is executed, if there is any.
ABLUnit project type

An ABLUnit project type is an OpenEdge project specialized for ABLUnit testing. The ABLUnit testing functionality allows you to write and execute the unit tests for the OpenEdge ABL code. For information on creating an OpenEdge ABLUnit project type, see Creating an ABLUnit project.

Tasks

Accessing ABLUnit perspective

The ABLUnit perspective helps you to create test class, test procedure, or test suite.

To access the ABLUnit perspective:

1. Do one of the following:
   - Click Open Perspective on the Workbench toolbar.
   - Select Window > Open Perspective.

2. Select Other > OpenEdge ABLUnit and click OK.

Creating an ABLUnit project

Using the New Project wizard, you can create an ABLUnit project that will contain all your test cases.

To create a new project:

1. Select File > New > OpenEdge Project from the Progress Developer Studio for OpenEdge main menu bar or the Project Explorer view context (right-click) menu. The New OpenEdge Project wizard opens and displays the Create an OpenEdge Project page on page 105.

2. Type a name in the Project name field.
   The root folder for the new project is created in your current Eclipse workspace, unless you specify a different location.

3. To choose a different location for the project, click and clear the Use default check box. Then either browse to, or enter the path to the desired location.

4. Select the General category from the left section.

5. Select the ABLUnit option button.

6. Click Next. The Select AVM and layout options page on page 107 appears.

7. Specify the OpenEdge AVM options and the Project layout options (or leave the default settings) and click Next. The Define PROPATH page on page 109 appears.
8. Leave the default settings or modify the PROPATH entries to be used by the project and click Next. The Select database connections page on page 109 appears.

9. Select the database connections to be used by the project and click Finish. A dialog appears with an option to open the new project in the ABLUnit perspective.

   A new project is created in the Project Explorer view. A tests folder is created under the new ABLUnit project that will contain the test cases.

Creating test cases using ABLUnit

A test case is a test class containing one or more test methods or one or more test procedures. By default when a test case is run, all contained test methods or test procedures are also run.

Progress Developer Studio for Open Edge provides the following ABLUnit wizards:

- New ABLUnit Test Class
- New ABLUnit Test Procedure
- New ABLUnit Test Suite Class
- New ABLUnit Test Suite Procedure

Note: You can create test cases or test suites using the ABLUnit wizards or create a new ABL Class or Procedure file, and manually enter the required test annotations.

To create a test case using the ABLUnit wizard:

1. Open an OpenEdge ABLUnit project in which you want to create a test case.
2. Select File > New. If the required test case or test suite does not appear in the menu, select Other. The New wizard appears.
3. Expand Progress OpenEdge > ABLUnit, and select the required test case or test suite type.
4. Click Next to launch the corresponding wizard.
5. Fill in the appropriate information for the new test case.
6. Click Finish.

   The test case of test suite you have created opens in the ABL Editor window.

Running test cases and test suites

You must have the [BLOCK | ROUTINE]-LEVEL ON ERROR UNDO, THROW statement in the test files to run the ABLUnit test cases.

Running test cases from OpenEdge

You can run an ABLUnit test case from the Progress Developer Studio for OpenEdge environment using the ABLUnit launch configuration. You can run the test cases or test suites or run a specific test method or test internal procedure inside a test case. It also supports running all the test cases and test suites in a specific project or package or folder. When you run a test case, all the test methods and test procedures defined in the test case are also run.
Accessing Progress OpenEdge ABLUnit launch configurations

To access the Progress OpenEdge ABLUnit launch configurations wizard, do one of the following:

- Click **Run** to run the currently selected ABLUnit test case.
- Click the drop-down arrow next to **Run** to open a menu from which you can start an existing configuration, create and run a default configuration.

The **Run Configurations** wizard opens.

**Defining and running a new ABLUnit launch configuration**

You can use the Run Configurations wizard to set all options for a new launch configuration or to edit an existing configuration.

**To define and run a new OpenEdge ABLUnit launch configuration:**

1. Click **Run Configurations** next to **Run**.
2. Click **New launch configuration** on the wizard toolbar and enter a name for the configuration in **Name**. The wizard creates a new ABLUnit launch configuration and displays it in the tree view in the left pane.
3. Set values as necessary for the available options. See Progress OpenEdge ABLUnit launch configuration settings for more information.
4. After defining the options, click **Apply** to save the configuration. You can also add this configuration to **Run** for later use. See *Adding an ABLUnit launch configuration to the Run menu* for more information.
5. To run the configuration, click **Run**. The ABLUnit test case runs. ABLUnit View opens and displays the test result.

**Adding an ABLUnit launch configuration to the Run menu**

It is helpful to add the launch configurations that you use frequently to the drop-down menu of **Run** on the main toolbar. This menu has a section (not visible when empty) at the top to which you can add any of your ABLUnit launch configurations.

**To add ABLUnit launch configuration:**

1. Click the drop-down arrow next to **Run** on the main toolbar.
2. Click **Organize Favorites**.

   The **Organize Run Favorites** dialog box opens.
3. Click **Add** to open the **Add Run Favorites** dialog box and select the ABLUnit launch configuration to be added to the menu.

   **Note:** The **Add Run Favorites** dialog box displays ABLUnit launch configurations only if they have been previously defined. See *Defining and running a new ABLUnit launch configuration* for more information.

   The ABLUnit launch configuration that you select in the **Add Run Favorites** dialog box displays in the **Favorites** section of the **Organize Run Favorites** dialog box.
4. Click **OK**.

**Copying or deleting ABLUnit launch configurations**

The Run Configurations wizard allows you to copy and delete existing ABLUnit launch configuration definitions.
To copy or delete an ABLUnit launch configuration:

1. Click the drop-down arrow next to Run on the main toolbar and click Run Configurations.
2. Do one of the following:
   - Right-click an ABLUnit launch configuration in the left pane and select Duplicate from the context menu. The wizard creates and opens a copy of the selected configuration. The duplicate configuration has the same name as the source, with an incremental number in parentheses appended.
   - Right-click an ABLUnit launch configuration in the left pane and select Delete from the context menu.

Running test cases from the command prompt

You can also run test cases and the test suites from the proenv command prompt. Run the ABLUnitCore.p driver program by passing the test case, the test suite, or the folder containing test cases and the test suites as a parameter.

You have the following options:

- Run multiple test cases by providing the test cases as parameters separated by commas (,...,...).
- Run only the test method or test procedure in a specific test case by providing the name of the test method or test procedure separated by # with the test case name.

The following command shows you how to use the ABLUnitCore.p driver program:

```plaintext
prowin32 -p (Driver Program) -param <parameters>
```

The following examples show you how to run a test class, a test procedure, a test suite, or a test folder:

```plaintext
prowin32 -p ABLUnitCore.p -param testpackage.testclass
prowin32 -p ABLUnitCore.p -param testprocedure.p
prowin32 -p ABLUnitCore.p -param testSuite.cls
prowin32 -p ABLUnitCore.p -param testSuite.p
prowin32 -p ABLUnitCore.p -param C:\tests
```

**Note:** You must provide the qualified name of the test class and the full path or the relative path of the test procedure.

To run the test classes, the classes must be available in the PROPATH.

The following is an example that passes multiple test cases:

```plaintext
prowin32 -p ABLUnitCore.p -param testclass1.cls
prowin32 -p ABLUnitCore.p -param testclass1#testmethod
prowin32 -p ABLUnitCore.p -param testclass1.cls,testclass2.cls
prowin32 -p ABLUnitCore.p -param testclass1#testM1,testclass2#testM2
prowin32 -p ABLUnitCore.p -param testProcedure1.p,testProcedure2.p
prowin32 -p ABLUnitCore.p -param testSuite.p prowin32 -p ABLUnitCore.p -param "C:\testFolder"
prowin32 -p ABLUnitCore.p -param testclass1.testclass
prowin32 -p ABLUnitCore.p -param testclass1.testclass#testmethod
prowin32 -p ABLUnitCore.p -param testclass1.testclass,testclass2.testclass	prowin32 -p ABLUnitCore.p -param testclass1#testM1,testclass2#testM2
prowin32 -p ABLUnitCore.p -param testProcedure1.p,testProcedure2.p
prowin32 -p ABLUnitCore.p -param testSuite.p prowin32 -p ABLUnitCore.p -param "C:\testFolder"
```
Note: Currently, we do not support running multiple tests from single test procedure or test class.

- Pass the configuration file as a parameter as shown below:

  prowin32 -p ABLUnitCore.p -param "CFG=C:\ABLUnit\tests\ablunit.json"

Here is a sample configuration (in JSON format) that displays all the valid options.

```json
{
  "options": {
    "output": {
      "location": "C:/ABLUnit/results",
      "format": "xml"
    },
    "quitOnEnd": true,
    "writeLog": true,
    "showErrorMessage": true,
    "throwError": true
  },
  "tests": [
    { "test": "C:/ABLUnit/tests/MyTestClass.cls" },
    { "test": "C:/ABLUnit/tests/MyTestProcedure.p" },
    { "folder": "C:/ABLUnit/tests/test-folder" },
    { "test": "C:/ABLUnit/tests/MyTestClass1.cls",
      "cases": [ "testItAgain" ]
    }
  ]
}
```
The options of the file are as follows:

- **options** specifies additional data for the run time framework.

- **output** specifies the location of the output directory to write the test results. If all the tests run successfully, an XML file that contains the results, **results.xml**, is created in the specified location (if output location is not specified, **results.xml** file is generated under the current working directory). You can change the location of results.xml by using the following command:

  `prowin32 -p ABLUnitCore.p -param "TestCaseClass.cls -outputLocation <path>"`

**Note:** Only xml results are supported and the only valid value for **format** is xml.

- The values for **quitOnEnd**, **writeLog**, **showErrorMessage**, **throwError** can be true or false only. If **quitOnEnd** is true then the application closes. It can be set to false, if another application uses the ABLUnit as a library. For example, if **writeLog** is true then a log file, **ablunit.log** is created in the current working directory and writes error messages to that log file. If **showErrorMessage** is set to true then the error messages is displayed in a new window. If ABLUnit is used as a library inside another program, set **throwError** to true, the framework displays the errors occurred.

- **tests** lets you add the tests that you want to run and each test can be configured using **test**.

- **folder** specifies the location that contain the tests, it is equivalent to using **test**, however the reverse is not true.

### Running test cases using an Apache ANT task

You can use a custom ANT task (build.xml) to run ABLUnit tests cases. The task is a part of ant-ablunit.jar and is located in:

%DLC%\java location

Make sure that Apache ANT is installed on your machine and copy the ant-ablunit.jar file into the installation location of Apache ANT.

You must write an XML file (build.xml) for ABLUnit that runs test cases from the ABLUnit testing framework and can display the results in the console or save it to an XML file.

You can define ABLUnit tests by nesting elements in the test cases or using batch tests to run the ANT task.

**ABLUnit Task-Level Properties**
You can set the following are properties that you can set at the ANT task level for ABLUnit. You can override the properties while running individual test or batch test elements.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLC</td>
<td>Specifies the installation location of OpenEdge.</td>
</tr>
<tr>
<td>environment (Optional)</td>
<td>Specifies the environment (GUI or TTY) in which test cases should run. By default, the test cases run in the GUI environment.</td>
</tr>
<tr>
<td>printsummary (Optional)</td>
<td>Prints a one-line summary of each test case that you run. The valid values are true, false, on, off, yes, and no. By default, the value is set to true.</td>
</tr>
<tr>
<td>haltonerror (Optional)</td>
<td>Stops the build process if an error occurs while running a test case. By default, the process proceeds even when errors occur.</td>
</tr>
<tr>
<td>haltonfailure (Optional)</td>
<td>Stops the build process if a test fails. By default, the process proceeds even if the test case fails.</td>
</tr>
<tr>
<td>tempdir (Optional)</td>
<td>Specifies the directory in which temporary files are placed while running the build process. By default, the working directory is used.</td>
</tr>
</tbody>
</table>

**ABLUnit Elements**

You can use the following ABLUnit elements in an ANT task to run an ABLUnit test case.

**dbinfo element**

Provides a database to be used while running a test case. This is a sub-element of the main ANT task and can appear any number of times. Each dbinfo element is considered as a single db connection. Set the following properties for this element:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the database name including the absolute path or the relative path if the database file is in the current working directory, for example in C:\OpenEdge\WRK114\sports2000.db.</td>
</tr>
<tr>
<td>host (Optional)</td>
<td>Specifies the host name of the server, for example, localhost.</td>
</tr>
<tr>
<td>port (Optional)</td>
<td>Specifies the port of the server or the service name, for example, 4545.</td>
</tr>
</tbody>
</table>

**Note:** The database must be available and the server must be active while running an ANT task.

**propath element**

Sets the PROPATH to be used while executing an ANT task. You can configure it with the pathelement element.

**Note:** For more information about pathelement, see the Apache ANT documentation.

**test element**
Configures an individual test case. Set the following properties for this element:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specifies the name of a test class (.cls) or a test procedure (.p) file.</td>
</tr>
<tr>
<td>todir (Optional)</td>
<td>Specifies the directory to which the results of the ANT task are written.</td>
</tr>
<tr>
<td>outfile (Optional)</td>
<td>Specifies the name of the test results file. The format attribute file decides the extension of the file.</td>
</tr>
<tr>
<td>format (Optional)</td>
<td>Specifies the format of the results. Currently, we support on XML format.</td>
</tr>
<tr>
<td>DLC (Optional)</td>
<td>Specifies the installation location of OpenEdge. This property overrides the value you set in the ABLUnit task.</td>
</tr>
<tr>
<td>environment (Optional)</td>
<td>Specifies the environment (GUI or TTY) in which test cases must run. It overrides the value you set in the ABLUnit task.</td>
</tr>
<tr>
<td>printsummary (Optional)</td>
<td>Prints a one-line summary of each test case that you run. The valid values are true, false, on, off, yes, and no. It overrides the value you set in the ABLUnit task.</td>
</tr>
<tr>
<td>haltonerror (Optional)</td>
<td>Stops the build process if an error occurs while running the test case. This property overrides the value you set in the ABLUnit task.</td>
</tr>
<tr>
<td>haltonfailure (Optional)</td>
<td>Stops the build process if a test case fails. This property overrides the value you set in the ABLUnit task.</td>
</tr>
</tbody>
</table>

**batchtest element**

Configures a batch test case. You can configure a batch test case with the fileset element.

**Note:** For more information about fileset, see the Apache ANT documentation.
Set the following properties for this element:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>todir</code> (Optional)</td>
<td>Specifies the directory into which the results of the ANT task are written.</td>
</tr>
<tr>
<td><code>format</code> (Optional)</td>
<td>Specifies the format of the test result.</td>
</tr>
<tr>
<td><code>DLC</code> (Optional)</td>
<td>Specifies the installation location of OpenEdge. This property overrides the value you set in the ABLUnit task.</td>
</tr>
<tr>
<td><code>environment</code> (Optional)</td>
<td>Specifies the environment (GUI or TTY) in which the test case must run. This property overrides the value you set in the ABLUnit task.</td>
</tr>
<tr>
<td><code>printsummary</code> (Optional)</td>
<td>Prints a one-line summary of each test case that you run. The valid values are true, false, on, off, yes, and no. This property overrides the value you set in the ABLUnit task.</td>
</tr>
<tr>
<td><code>haltonerror</code> (Optional)</td>
<td>Stops the build process if an error occurs while running the test case. This property overrides the value you set in the ABLUnit task.</td>
</tr>
<tr>
<td><code>haltonfailure</code> (Optional)</td>
<td>Stops the build process if a test fails. This property overrides the value you set in the ABLUnit task.</td>
</tr>
</tbody>
</table>

Here is a sample build.xml file (Ant task):

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project name="ABLUnit-Ant" default="main" basedir=".">

  <!-- ABLUnit task definition target starts here -->
  <target name="taskdef">
    <taskdef name="ablunit" classname="com.progress.openedge.ant.ablunit.ABLUnitTask"
             classpath="dist/ant-ablunit.jar" />
  </target>

  <!-- Main target starts here -->
  <target name="main" depends="taskdef">
    <ablunit dlc="C:\Progress\OpenEdge114" environment="gui"
             printsummary="true" haltonerror="no" haltonfailure="no" tempdir="${basedir}/tmpdir">
      <dbinfo name="C:\OpenEdge\WRK114\s2000.db" host="localhost" port="4545" />
      <propath>
        <pathelement location="${basedir}/samples/classes" />
        <pathelement location="${basedir}/samples1/classes" />
      </propath>
      <test name="${basedir}/samples/MyTestClass.cls" todir="${basedir}/results"
            outfile="result-MyTestClass-gui" format="xml" />
      <test name="${basedir}/samples/MyTestProcedure.p" todir="${basedir}/results"
            outfile="result-MyTestProcedure" format="xml" />
      <batchtest todir="${basedir}/results" format="xml">
        <fileset dir="${basedir}/samples1">
          <include name="**/*.cls" />
          <include name="**/*.p" />
        </fileset>
      </batchtest>
    </ablunit>
  </target>
</project>
```
Using the ABLUnit view

You can use the ABLUnit view to:

- Display the progress and status of a running test case
- Import results of an executed test case from an XML file
- Open the corresponding file that contains the test case
- View details of the last ten executed test cases and run those test cases

The ABLUnit view is displayed in the ABLUnit perspective by default. If it is not already displayed, you can access it by selecting Window > Show View > Progress OpenEdge Editor > ABLUnit.

To open the file containing the test cases, do one of the following:

- Select Go to File from the context (right-click) menu of the ABLUnit view
- Double-click the file in the tree view of the ABLUnit view

The corresponding file opens in an editor.

To view details of the last ten executed test cases and to run any of those test cases, do the following:

1. Select the drop-down arrow of the Test Run History control on the ABLUnit view toolbar.
   
   The drop-down list displays the name of the test case (path relative to the workspace) followed by the date and time (yyyy/mm/dd hh:mm:ss) when the test was executed.

2. Double-click to select the test case that you want to execute from the list.

Note: For more information about the controls available in the ABLUnit view, see ABLUnit view on page 261.

Reference

Test case generation wizards

This section includes the following topics:

New ABLUnit Test Class wizard

The New ABLUnit Test Class wizard allows you to create an ABLUnit test class. You can also select the class for which the test methods should be generated.

This wizard appears when you select File > New > ABLUnit > Test Case Class from main menu or from New > ABLUnit > Test Case Class from the context menu in the Project Explorer view.
The following controls are available in this wizard:

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package root</td>
<td>(Required) Specifies a currently open project that will contain the class code, other project code, and any assembly references. Click Browse if you want to select a project other than the current one (the default value).</td>
</tr>
<tr>
<td>Package</td>
<td>Optionally, specifies a package name that will contain the class file, corresponding to a sub-folder of the package root. Click Browse or enter the folder path. A period (.) must separate each subfolder from its parent folder in the path name.</td>
</tr>
<tr>
<td>Class name</td>
<td>(Required) Specifies the name of the test case class. The name must begin with a letter; spaces and most non-alphanumeric characters are not allowed. The .cls extension is appended automatically.</td>
</tr>
<tr>
<td>Inherits</td>
<td>Optionally specifies another class in the current project as a super class from which the new class inherits state and behavior. Click Browse or enter the class path by typing. A period (.) must separate each node from its parent folder in the path name.</td>
</tr>
<tr>
<td>Note:</td>
<td>The super class cannot be generic.</td>
</tr>
</tbody>
</table>
| Which method stubs would you like to create? | Optionally, allows you to generate stubs for any of the following methods:  
  • Before - Adds a test method annotated using @Before to the test class  
  • After - Adds a test method annotated using @After to the test class.  
  • Setup - Adds a test method annotated using @Setup to the test class.  
  • TearDown - Adds a test method annotated using @TearDown to the test class. |
| Class to test        | Specifies the class for which you want to create the ABLUnit test class. Click Browse to view all the available classes in the selected project or enter the class path by typing. A period (.) must separate each node from its parent folder in the path name. |
| Next                 | Displays the Select Methods page that lists all the methods in the selected class. On this page, you can select the methods for which you want the stubs to be generated. |
| Finish               | Generates a test class with the selected test method stubs and opens it in the ABL Editor.                                                   |
New ABLUnit Test Case Procedure wizard

The New ABLUnit Test Procedure wizard allows you to create a test procedure (.p) file. Additionally you can select the procedure for which the test procedures should be generated.

This wizard appears when you select **File > New > Test Case Procedure** from the main menu or **New > Test Case Procedure** from the context menu of the **Project Explorer** view.

The following controls are available:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Container</td>
<td>Specifies the currently open project or a folder within the project where the test procedure is to be created. Click <strong>Browse</strong> if you want to select a project other than the current one.</td>
</tr>
<tr>
<td>File name</td>
<td>(Required) Specifies the test procedure name. The .p extension is appended automatically.</td>
</tr>
<tr>
<td>Setup</td>
<td>Adds a test internal procedure annotated using @Setup in the test procedure.</td>
</tr>
<tr>
<td>TearDown</td>
<td>Adds a test internal procedure annotated using @TearDown in the test procedure.</td>
</tr>
<tr>
<td>Before</td>
<td>Adds a test internal procedure annotated using @Before in the test procedure.</td>
</tr>
<tr>
<td>After</td>
<td>Adds a test internal procedure annotated using @After in the test procedure.</td>
</tr>
<tr>
<td>Procedure to test</td>
<td>Specifies the ABL procedure for which you want to create the test procedure. Click <strong>Browse</strong> if you want to select a procedure file other than the current file.</td>
</tr>
<tr>
<td>Finish</td>
<td>Generates a test procedure with the selected annotations, which opens in the ABL Editor.</td>
</tr>
</tbody>
</table>

New ABLUnit Test Suite Class wizard

The New ABLUnit Test Suite Class wizard allows you to create a test suite class. A test suite class contains a list of test cases and/or other test suites. You can use test suites for grouping and organizing test cases. When you run a test suite, all the test cases and suites that belong to that test suite are executed by default.

This wizard appears when you select **File> New > ABLUnit > Test Suite Class**. You can also select **New > ABLUnit > Test Suite Class** from the context menu in the Project Explorer view.

The following controls are available in this wizard:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package root</td>
<td>(Required) Specifies a currently open project that will contain the class code, other project code, and any assembly references. Click <strong>Browse</strong> if you want to select a project other than the current one (the default value).</td>
</tr>
</tbody>
</table>
### New ABLUnit Test Suite Procedure wizard

The New ABLUnit Test Suite Procedure wizard allows you to create a test suite procedure. A test suite procedure contains one or more test cases. You can select both the test class and test procedure files as part of your test suite procedure.

When a test suite procedure is run, all the test cases included in the test suite procedure are executed.

This wizard appears when you select **File > New > Test Suite Procedure**. You can also select **New > ABLUnit > Test Suite Class** from the context menu in the **Project Explorer** view.

The following controls are available:

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Container</strong></td>
<td>Specifies the currently open project or a folder within the project where the test procedure is to be created. Click Browse if you want to select a project other than the current one (the default value).</td>
</tr>
<tr>
<td><strong>File name</strong></td>
<td>(Required) Specifies the test suite procedure name. The .p extension is appended automatically.</td>
</tr>
<tr>
<td><strong>Add</strong></td>
<td>Opens the Select Class or Procedure dialog which allows you to select the test class or test procedure files which you want to include in the suite.</td>
</tr>
</tbody>
</table>

Table: Description of fields

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
<td>Optionally, specifies a package name that will contain the class file, corresponding to a sub-folder of the package root. Click Browse or enter the folder path. A period (.) must separate each subfolder from its parent folder in the path name.</td>
</tr>
<tr>
<td>Class name</td>
<td>(Required) Specifies the name of the test suite class. The name must begin with a letter; spaces and most non-alphanumeric characters are not allowed. The .cls extension is appended automatically.</td>
</tr>
<tr>
<td>Test classes and procedures to include in the suite</td>
<td>Displays a list of test classes and procedures that you add to or remove from the test suite.</td>
</tr>
<tr>
<td>Add</td>
<td>Opens the Select Class or Procedure dialog that allows you to select test class and test procedure files that you want to include in the test suite.</td>
</tr>
<tr>
<td>Remove</td>
<td>Deletes an item from the list of test classes and procedures to be included in the test suite class.</td>
</tr>
<tr>
<td>Finish</td>
<td>Generates a test suite class with the selected test classes and procedures included and opens it in the ABL Editor.</td>
</tr>
<tr>
<td>Field name</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Remove</td>
<td>Deletes a selected test class or test procedure file from the test suite procedure.</td>
</tr>
<tr>
<td>Finish</td>
<td>Saves your selections and generates a test suite procedure file, which opens in the ABL Editor.</td>
</tr>
</tbody>
</table>

**ABLUnit view**

The ABLUnit view displays the test run progress and status, and also allows you to import a test run result from a XML file.

This view includes the following controls:

<table>
<thead>
<tr>
<th>Controls</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![rere run icon]</td>
<td>Reruns a particular test method, test class, test procedure, or test suite.</td>
</tr>
<tr>
<td>![next failed test icon]</td>
<td>Shows the next failed test.</td>
</tr>
<tr>
<td>![previous failed test icon]</td>
<td>Shows the previous failed test.</td>
</tr>
<tr>
<td>![failed cases icon]</td>
<td>Shows failed cases only.</td>
</tr>
<tr>
<td>![import test case icon]</td>
<td>Imports the results of a test case that was previously run.</td>
</tr>
<tr>
<td>![display test case details icon]</td>
<td>Displays details of the last ten executed test cases and lets you execute them.</td>
</tr>
<tr>
<td>![export test case icon]</td>
<td>Exports the results of a test case, which was previously run, to an XML file.</td>
</tr>
</tbody>
</table>

**Progress OpenEdge ABLUnit launch configuration settings**

The ABLUnit launch configuration wizard displays the following tabs:

<table>
<thead>
<tr>
<th>Tab name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABLUnit</td>
<td>Lets you run a single test case or multiple test cases from the folder.</td>
</tr>
<tr>
<td>Startup</td>
<td>Lets you specify the startup parameters for running test cases.</td>
</tr>
<tr>
<td>PROPATH</td>
<td>Lets you control where the AVM searches for files and procedures.</td>
</tr>
<tr>
<td>Databases</td>
<td>Lets you add the database connection for running the configuration.</td>
</tr>
<tr>
<td>Common</td>
<td>Lets you set options for specifying where an ABL application launch configuration is saved, how it is accessed, and how it runs.</td>
</tr>
</tbody>
</table>
Progress OpenEdge ABLUnit debug configuration settings

The ABLUnit debug configuration wizard displays the following tabs:

<table>
<thead>
<tr>
<th>Tab name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABLUnit</td>
<td>Lets you debug a single test case or multiple test cases from the folder.</td>
</tr>
<tr>
<td>Startup</td>
<td>Lets you specify the startup parameters for running test cases.</td>
</tr>
<tr>
<td>PROPATH</td>
<td>Lets you control where the AVM searches for files and procedures.</td>
</tr>
<tr>
<td>Databases</td>
<td>Lets you add the database connection for running the configuration.</td>
</tr>
<tr>
<td>Common</td>
<td>Lets you set options for specifying where an ABL application debug configuration is saved, how it is accessed, and how it runs.</td>
</tr>
</tbody>
</table>

Add New Test Method wizard

The Add New Test Method wizard enables you to add a test method to a test class (.cls) file when the file is open in ABL editor.

To access the wizard, select Source > Add Test Method from the context menu in ABL editor or from the main menu. You can also click Alt+Shift+L key sequence.

**Note:**
This option is enabled only when you have a test class (.cls) file open in ABL editor. This option is visible only when the current perspective is ABLUnit perspective or the class file is a part of ABLUnit faceted project.

The wizard has the following controls that you can use to add a test method to a .cls file:

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method name</td>
<td>Specifies a name for the test method that you want to add to your required test class file.</td>
</tr>
<tr>
<td>Annotation</td>
<td>Specifies an annotation type for the test method from the drop-down menu. The annotation types are Test, Before, After, Setup, and TearDown.</td>
</tr>
<tr>
<td>Insertion position</td>
<td>Specifies the position in current test class file where you want to place the annotated test method.</td>
</tr>
<tr>
<td>Generate</td>
<td>Generates the annotated test method in your test class file.</td>
</tr>
</tbody>
</table>
Add New Test Procedure wizard

The Add New Test Procedure wizard enables you to add a test internal procedure to a test procedure file when the file is open in ABL editor.

To access the wizard, right-click the file and select Source> Add Test Procedure > from the context menu in ABL editor or the main menu. You can also click the Alt+Shift+K key sequence.

**Note:** This option is enabled only when you have a test procedure file open in ABL editor. This option is visible only when the current perspective is ABLUnit perspective or the procedure file is a part of ABLUnit faceted project.

The wizard has the following controls that you can use to add a test procedure to a file:

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Procedure name</td>
<td>Specifies a name for the test internal procedure that you want to add to your required test procedure file.</td>
</tr>
<tr>
<td>Annotation</td>
<td>Specifies an annotation type for the test internal procedure from the drop-down menu. The annotation types are Test, Before, After, Setup, and TearDown.</td>
</tr>
<tr>
<td>Insertion position</td>
<td>Specifies the position in your test procedure file where you want to place the annotated test internal procedure.</td>
</tr>
<tr>
<td>Generate</td>
<td>Generates the annotated test internal procedure in your test procedure file.</td>
</tr>
</tbody>
</table>

Assert API

The Assert API provides a set of assertion methods to validate the expected result of a method or procedure in a test with that of the actual result.


**Note:** All the Object-related APIs are a part of the OpenEdge.Core.Assertion.AssertObject class in the OpenEdge.Core.pl file.

**Note:** For more information on a detailed list and description of the classes and the methods, see [https://documentation.progress.com/output/oehtpcient/index.html](https://documentation.progress.com/output/oehtpcient/index.html).
Chapter 6: Overview of ABLUnit testing framework
Data Object overview

An OpenEdge Data Object implements a server-side resource that provides OpenEdge data and operations for access by a resource client through the agency of a Data Object Service. This resource is implemented by an ABL Business Entity, which is typically an ABL user-defined class that is coded to access the data through methods that implement the resource operations.

A Data Object Service is a REST Web service and a Data Service Catalog. This Data Service Catalog defines the schema of the data and the operations on the data that is supported by one or more Data Object resources. In this way, a single Data Object Service can support client access to multiple resources (Data Objects) running on a single OpenEdge application server.

Progress Developer Studio for OpenEdge provides wizards and other tools to accelerate the development of server-side Data Objects and Data Object Services.

Progress Developer Studio for OpenEdge allows you to develop and deploy Data Object Services that you can use to provide OpenEdge data to mobile apps running in native device environments or to web apps, which can run in any browser-based environment.

You can develop Data Objects Services using:

- Data Object projects that support Data Object Service development using facets that work with the OE AppServer and a Tomcat web server that hosts the Data Object Services.

- ABL Web App projects that support Data Object Service development using facets that work with PAS for OpenEdge only. For more information on working with the ABL Web App project and creating Data Object Services, see the Progress Application Server for OpenEdge help.
For more information on how mobile and web apps can use OpenEdge Data Objects to access OpenEdge data and business logic, see the Progress Data Objects Guide and Reference documentation at the https://documentation.progress.com/output/pdo location.

For details, see the following topics:

- **Concepts**
- **Tasks**
- **Reference**

**Concepts**

**Data Object overview**

An OpenEdge Data Object implements a server-side resource that provides OpenEdge data and operations for access by a resource client through the agency of a Data Object Service. This resource is implemented by an ABL Business Entity, which is typically an ABL user-defined class that is coded to access the data through methods that implement the resource operations.

A Data Object Service is a REST Web service and a Data Service Catalog. This Data Service Catalog defines the schema of the data and the operations on the data that is supported by one or more Data Object resources. In this way, a single Data Object Service can support client access to multiple resources (Data Objects) running on a single OpenEdge application server.

Progress Developer Studio for OpenEdge provides wizards and other tools to accelerate the development of server-side Data Objects and Data Object Services.

Progress Developer Studio for OpenEdge allows you to develop and deploy Data Object Services that you can use to provide OpenEdge data to mobile apps running in native device environments or to web apps, which can run in any browser-based environment.

You can develop Data Objects Services using:

- Data Object projects that support Data Object Service development using facets that work with the OE AppServer and a Tomcat web server that hosts the Data Object Services.
- ABL Web App projects that support Data Object Service development using facets that work with PAS for OpenEdge only. For more information on working with the ABL Web App project and creating Data Object Services, see the Progress Application Server for OpenEdge help.

**Prerequisites for Data Object project support**

The following are the prerequisites for Data Object support in the Progress Developer Studio for OpenEdge:
• OpenEdge Explorer (OEE) to work with the Server functionality. By default, OpenEdge Explorer is shipped with OpenEdge. If you have not configured OEE during OpenEdge installation, you can manually configure it post installation. For more information, see the OpenEdge Getting Started: Installation and Configuration guide.

• Java Runtime Environment (JRE) v1.7. By default, JRE v1.7 is shipped with OpenEdge.

**See also**
Data Object overview on page 265

### Terminology related to Data Object support

The following terms describe various components that apply to Data Object support in Progress Developer Studio for OpenEdge:

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABL routine</td>
<td>An ABL routine can be internal procedures, user-defined functions, or class methods.</td>
</tr>
<tr>
<td>Apache Tomcat</td>
<td>A Java container provided with OpenEdge that allows you to deploy Data Object services. When installing Progress Developer Studio for OpenEdge, Apache Tomcat is also installed and configured. The Tomcat Java container will be located in your OpenEdge installation directory at $DLC/servers/tomcat.</td>
</tr>
</tbody>
</table>
| JavaScript Data Object (JSDO)             | Progress-supported, JavaScript implementation of the open source Cloud Data Object. It provides client access to the data and operations of a single Data Object resource, so the client code can call methods on a JSDO to execute the Data Object operations on the server. The data for these operations is serialized between the Data Object Service and the client as JSON media. The JSDO uses a Data Service Catalog file to acquire the client resource's metadata from the server. For more information on the Cloud Data Object open source project, see [http://clouddataobject.github.io/](http://clouddataobject.github.io/) and for detailed information on the JSDO, see the documentation on Progress Data Objects at [https://documentation.progress.com/output/pdo/](https://documentation.progress.com/output/pdo/). For more information, see:  
  • Creating a Business Entity class  
  • Data Service Catalog schema properties |
<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
</table>
| Data Service Catalog file | A Data Service Catalog file is generated from a Business Entity class file in Progress Developer Studio for OpenEdge. It provides the definition of the resources in a Data Object service. The Data Service Catalog includes an entry for each built-in (CRUD and Submit) method and each invocation (or Invoke) method for a resource. It also includes the schema for the resource. The JSDO uses the schema information for its built-in CRUD and Submit methods. For each method, the catalog provides parameter mapping information. This information (metadata) from the catalog is used to construct the HTTP request for the operation calls made to the REST adapter. For more information, see:  
  • Creating a Business Entity class  
  • JSDO Catalog schema properties |
| Business Entity | A Business Entity defines a resource that contains schema and operations (business logic) to work on the schema. It contains pre-defined Data Object interface annotations for the schema and operations. The Data Object interface annotated Business Entity class is exposed as a Data Object resource.  
  An OpenEdge Data Object uses the REST Adapter as an interface to the OpenEdge AppServer to run the Business Entity’s operations. A Business Entity follows a prescriptive programming model that works well with the JavaScript Data Object (JSDO). It can be exposed directly to a JSDO or as a separate service interface.  
  For information on creating Business Entity class files, see:  
  • Creating a Business Entity class  
  • New Business Entity wizard |
<p>| Data Object resource | A Business Entity or other singleton class or procedure that has a Data Object interface defining a data model (a single temp-table or ProDataSet) and Data Object operations. A Business Entity as defined by the OpenEdge Reference Architecture is an example of a resource. It is expected to define a data using logical schema along with business logic to access and update the data. It is always an abstraction from the physical database. This can also be considered the back end of a remote data object. For more information, see Defining Data Object resources. |</p>
<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Object Service</td>
<td>Data Object Service is the service interface for a resource. Resources within a Data Object service are accessed using a client side API. OpenEdge provides the JSDO, which provides this API for JavaScript clients. Each resource is defined in a catalog file which identifies the schema and operations supported for the resource. The Data Object Services defined for your project appear under the Defined Services node and the .json files appear under the WebContent node in your project directory in the Project Explorer view. For more information, see Creating a Data Object Service on page 280.</td>
</tr>
<tr>
<td>Data Object Web Application (WAR)</td>
<td>A Data Object web application is one or more Data Object Services packaged for deployment to the Apache Tomcat Java container in a war format. For more information, see the Publishing Data Object Services on page 286 and Exporting a Data Object Web Application on page 287.</td>
</tr>
<tr>
<td>OE Web Server</td>
<td>A web server that enables you to deploy, configure, and manage Data Object Services in additional to OE REST applications. The OE Web server application enables you to deploy and manage Data Object Services on the Web server. For more information, see Creating an OE Web Server instance on page 285.</td>
</tr>
</tbody>
</table>

**See also**

OpenEdge Data Object overview on page 265

### OpenEdge Data Object project type

A Data Object project type is an OpenEdge project that is specialized for Data Object Service development and deployment. An OpenEdge Data Object project contains the following OpenEdge facets:

- Data Object
- REST
- AppServer
- Progress Adapters
- Static Web content
- JavaScript

You can see the project facets that are associated with an Data Object project using the Project Facets properties page. To access Project Facets properties page, select Properties from the OpenEdge Data Object project right-click menu. For more information, see page.

**Note:** Data Object projects can be deployed only to traditional servers not to PAS for OpenEdge.
See also

Creating an OpenEdge Data Object project on page 274

Data Object project folder structure

The New OpenEdge Project wizard enables you to create Data Object projects in Progress Developer Studio for OpenEdge. Each Data Object project you create contains the following nodes:

- **.services**
  Contains the artifacts of the exposed Data Object and REST services. By default, the .services folder contains empty Consume and Expose nodes. On generation of a Data Object service, its artifacts are generated and listed under the Expose node. For more information, see Data Object service artifacts on page 271.

- **.settings**
  Contains the facets information that are installed for the project and the server related configuration.

- **AppServer**
  Contains the files that you want to publish to the server. A default node with the name AppServer is created when you define an OpenEdge Data Object project.

- **RESTContent**
  Contains the files that will be used for generating Data Object service (WAR) file while packaging and deploying. This RESTContent folder is shared by all the Data Object and REST applications in the project.

  **Note:** The content of the RESTContent folder is copied from your OpenEdge installation directory $DLC\rest\server.

- **WebContent**
  The .json file is created on defining a Data Object Service in the WebContent folder. A separate .json file is created for each defined Data Object service with the same name as that of the Data Object service name.

  **Note:** The .json file is created in the WebContent folder on defining a Data Object service. A separate .json file is created for each defined Data Object service with the same name as that of the Data Object service name.

- **Defined Services**
  Contains a list of defined Data Object and REST services. A default Data Object service with the project name and Service appended to it is created when you define an OpenEdge Data Object project. For example, <projectname>Service.

**Note:** The .services, and .settings node do not appear by default under your project directory in the Project Explorer view. To view these nodes, open Available Customizations dialog (select View Menu > Customize view from the Project View toolbar), clear the .*resources check box, and then click OK. Progress recommends you to not edit the files listed under these hidden nodes.

See also

Creating a Data Object project on page 274
OpenEdge Data Object project type on page 269
Data Object application WAR file structure on page 271
Creating a Data Object service on page 280
Data Object service artifacts

The Data Object services you define using the New Data Object Service wizard appears under the Defined Services node in your project.

The Data Object artifacts are a combination of REST artifacts followed by Data Object. The Data Object service artifacts are generated under .services > Expose > rest > service_name in your project directory, wherein service_name is the name of the defined Data Object service. A separate Data Object service node is created for each Data Object service you define, and its artifacts are generated under that Data Object service node.

Note: The .services node does not appear by default under your project in the Project Explorer view. To view the .services node, open Available Customizations dialog (select View Menu > Customize view from the Project View toolbar), clear the .*resources checkbox, and then click OK.

On creation of a Data Object service, the following artifacts are generated:

.PIDL files
Generates .pidl files for the ABL class and procedure files that you select while creating a Data Object service. The .pidl files are listed under the .services > AppServer node in your project directory.

resourceModel.xml
The file contains resource URI and mapping information of the verbs with operations (Data Object interface annotated ABL routines). The resourceModel.xml file is listed under .services > Expose > rest > service_name in your project directory.

spring.xml
The file contains the AppServer end point details and resourceModel.xml entry. The spring.xml file is listed under .services > Expose > rest > service_name in your project directory.

<service_name>.json
The file contains information about the operations and schema for Data Object resources. The .json file is a replica of the selected Data Object resources (ABL class and procedure files with Data Object interface annotations) that is represented in the JSON (Java Script Object Notation) format. The .json files are listed under the WebContent directory with serviceName.json in your project directory.

Note: Progress recommends you to not modify or delete a Data Object service artifact or any other file listed under the .services node.

See also
OpenEdge Data Object project type on page 269
Creating a Data Object service on page 280

Data Object application WAR file structure

The below table describes the directory structure of a Data Object Web application (WAR file):

<table>
<thead>
<tr>
<th>Folder/File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/</td>
<td>The Data Object application (web application) root directory. This directory contains META-INF, static, WEB-INF, and index.html.</td>
</tr>
<tr>
<td>/META-INF</td>
<td>Contains only the MANIFEST.MF file.</td>
</tr>
<tr>
<td>Folder/File</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>/static</td>
<td>Contains the static HTML pages, images (gif, jpeg), style sheets (css), and any other static files you want to locate on the Web server. Also contains Data Service catalog (.json) file.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: The Data Service catalog .json file is copied from the WebContent node in your Data Object project directory.</td>
</tr>
<tr>
<td>/WEB-INF</td>
<td>Contains the configuration files for the Data Object application. This directory contains adapters, classes, lib, web.xml, appSecurity-xxxxxx.xml files, and mvc-dispatch-content.xml.</td>
</tr>
<tr>
<td>/index.html</td>
<td>This HTML page is an OpenEdge supplied template welcome page.</td>
</tr>
<tr>
<td>/static/auth</td>
<td>Contains HTML files for use in logging into and out of Data Object application. The files login.html, logout.html, and loginfail.html are OpenEdge supplied template pages that support the HTTP form-authentication model.</td>
</tr>
<tr>
<td>/static/error</td>
<td>Contains OpenEdge supplied HTML template pages for customizing HTTP error returns.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: This will override the Java containers default error pages and provide a consistent output for all Java containers.</td>
</tr>
<tr>
<td>/static/home.html</td>
<td>This HTML page is an OpenEdge supplied HTML template landing page for successful log in and log out operations.</td>
</tr>
<tr>
<td>/WEB-INF/adapters</td>
<td>Contains the adapter application runtime properties runtime.props file and service .paar files.</td>
</tr>
<tr>
<td>/WEB-INF/classes</td>
<td>Contains the class files used by the Data Object Services. This is where .class files are loaded from when the Data Object application is executing. Also, contains the REST adapter log log4j.properties file.</td>
</tr>
<tr>
<td>/WEB-INF/lib</td>
<td>Contains the runtime library files. The library files are copied from your OpenEdge installation directory $DLC/rest/lib.</td>
</tr>
<tr>
<td>/WEB-INF/appSecurity-xxxxxx.xml</td>
<td>The appSecurity-xxxxxx.xml files are OpenEdge supplied template security configurations for common authentication models.</td>
</tr>
<tr>
<td>/WEB-INF/mvc-dispatch-context.xml</td>
<td>This file is OpenEdge supplied template configuration file for spring MVC dynamic page generation.</td>
</tr>
</tbody>
</table>
### See also

Exporting as a Data Object Web Application WAR file on page 287

### Tasks

#### Setting default preferences for starting servers automatically

By default, Developer Studio does not start the servers when you open Developer Studio. The **Startup** preferences page lets you change this setting by specifying the default servers and setting the servers to start automatically when you open Developer Studio.

To set preferences for starting servers automatically:

1. Select Window > Preferences > Progress OpenEdge > Startup.
2. In the Default servers section, select the AppServer (as `restbroker1`) and the OE Web Server (as `restmgr1`) that you want to start automatically.
3. Select Start AppServer and the OE Web Server automatically check box to start the servers by default when you start the Developer Studio.
4. Click Apply and then click OK. The preferences are set.
Working with Data Objects

Progress Developer Studio for OpenEdge allows you to create the Data Object Services using one of the following ways to build a Data Object application:

• Create a customized Data Object application following a series of steps and wizards.

  This approach involves the following tasks:

  • Creating an OpenEdge Data Object project

    Note: If you wish to deploy the Services to PAS for OpenEdge server, create an ABL Web App project.

  • Defining Data Object resources
  • Creating a Data Object Service

• Build and deploy, automatically, a complete Data Object Service instantly.

  This approach involves the following task:

  Creating an express Data Object Service

Creating a Data Object project

You can create an OpenEdge Data Object project and customize the options for Data Object services development.

Note: You can also choose to build and deploy a Data Object Service. For information on automatically building and deploying a Data Object Service, see Creating an express Data Object application.

To create an OpenEdge project specialized for Data Object Service development:

1. Select File > New > OpenEdge Project from the Progress Developer Studio for OpenEdge main menu bar or the Project Explorer view context (right-click) menu. The New OpenEdge Project wizard opens and displays the Create an OpenEdge Project page on page 105.

2. Type a name in the Project name field.

   The root folder for the new project is created in your current Eclipse workspace, unless you specify a different location.

3. To choose a different location for the project, click and clear the Use default check box. Then either browse to, or enter the path to the desired location.

4. Select Server from the left section.

5. Select Classic option button to select the server type that you want to deploy the application to.

6. Select REST as the transport.

7. Select the Create Data Object Service check box.

   Note: If you want to create a Data Object project with Express setup, select the Express setup check box and click Finish.

8. Click Next. The Select AVM and layout options page appears.
9. Specify the **OpenEdge AVM** options and the **Project layout** options (or retain the default settings). For more information, see the page.

10. Click **Next**. The **Define AppServer content module** page appears.

11. In the **Module name** field, enter the name for the AppServer module. The default is the project name.

12. In the **AppServer source folder** field, enter a name of the AppServer module content directory. The default is a folder named AppServer.

13. From the **Supported servers** list, select a broker that will be the destination for publishing your AppServer application service files. The primary function of these servers is to support publishing your code for running, testing, and deploying your Data Object service.

   **Note**: Data Object projects can be deployed only to traditional servers not PAS for OpenEdge.

14. Click **Next**. The **New Data Object Service** page appears.

15. Select the **Create a Data Object service** check box to create a default Data Object service for the Data Object project and associate the Data Object service with the OE Web Server instances. For more information, see the Creating a Data Object Service on page 280 wizard.

16. Click **Next**. The **Define PROPATH** page appears.

17. Retain the default settings or modify the PROPATH entries to be used by the project. For more information , see the appears.

18. Click **Next**. The **Select database connections** page appears.

19. Select the database connections to be used by the project. For more information, see the page.

20. Click **Next**. The **Static Web Project** page appears.

21. Configure settings for the static Web module. For more information, see the page.

22. Click **Finish**.

You will see the OpenEdge Data Object project created in the **Project Explorer** view. Open the OpenEdge Data Object project in the OpenEdge Server perspective. On successful creation of an OpenEdge Data Object project, the AppServer, REST, Progress Adapters, JavaScript, Data Object, and Static Web Module facets are associated for a Data Object project. The **Project Facets** page displays the facets associated with your current OpenEdge project. For more information, see the page.

**See also**

- [OpenEdge Data Object project type](#) on page 269
- [Data Object project folder structure](#) on page 270
- [Creating an Express Data Object application](#) on page 283

**Defining Data Object resources**

**Creating a Business Entity class**

A Business Entity class file with Data Service Catalog schema properties can be generated through Progress Developer Studio for OpenEdge by either creating an express Data Object project or by using the **New Business Entity** wizard.

This section explains using the **New Business Entity** wizard to create a Business Entity class.
The Business Entity class files generated in Progress Developer Studio for OpenEdge inherit from the abstract class `OpenEdge.BusinessLogic.BusinessEntity`, which contains generic methods performing the CRUD and Submit operations.

Depending on how you create a Business Entity class, Developer Studio either generates default code or generates only stubs for the methods that implement operations for the Data Object resource. In any case, you may need to add or modify any default code to implement certain features. For information on how to add and modify code for a Business Entity class, see the sections on Data Object Services in *OpenEdge Development: Web Services* guide.

The **New Business Entity** wizard provides the following ways to create a Business Entity class:

- From a selected database table
- From the schema of a selected source file

However, when generating the Data Service Catalog file, Progress Developer Studio for OpenEdge uses the Abstract Syntax Tree (AST) to read the DataSet and temp-tables in the Business Entity class file, regardless of how it was created (from a database table or using the schema from a source file).

**See also**
- OpenEdge Data Object project type on page 269
- Creating a Data Object project on page 274
- New Business Entity wizard on page 293

### Creating a Business Entity class from a database table

To create a Business Entity class from a selected database table:

1. Select an OpenEdge Data Object project and select **New Business Entity** from the context menu.
   
   The *Create a Business Entity class* page of **New Business Entity wizard** appears.

2. Specify values for the controls on the *Create a Business Entity class* page. See *Create a Business entity class page* on page 293 for detailed information about each control.

3. Click **Next**.
   
   The *Select a schema file* page appears.

4. The **Resource name** field displays the name specified in the **Business Entity name** field. You can enter a different name. This is an alias of Business Entity class name.

5. Select the operation as **Read-only**, **CRUD**, or **CRUD and Submit**. The ABL code is generated with the specified operations.

6. Select **Write dataset before image**, if you want the before-image data written to the file.

   **Note:** The **Write dataset before image** option is selected by default if the **CRUD and Submit** option is selected. The before-image data is required to perform the Submit operation.

7. Select the **Select database table** option button to select a database connection and table.
   
   a) In the **Connection** field, select the database connection that you want to use to dynamically define the temp-table.
   
   b) In the **Table** field, select the database table from selected database connection.

8. Select **Expose as Data Object Service** to expose the defined Business Entity class as a Data Object resource.
Note: This generates default Data Object interface annotations to define a Business Entity as a Data Object resource.

9. Specify a Uniform Resource Identifier (URI) to access the Data Object resource in the Resource URI field. By default, the resource name is displayed in the field with a prefix "/". You can enter a different URI name.

10. Click Finish.

The wizard creates and opens the Business Entity class, which includes the temp-table definition with the Data Object interface (described in the Data Service Catalog) annotations defining the scheme attributes, in the ABL Editor.

Note: You can manually edit or add annotations to the Business Entity class. For more information about these annotations, see JSDO Data Service schema properties.

The wizard also generates a Data Object resource from the Business Entity class file. By default, the defined Business Entity class files are listed under the AppServer node in your project directory.

Note: When you add a Business Entity class file to a non-Data Object project such as OpenEdge or AppServer project except ABL Web App project, the Data Object facet gets enabled for the project. The project behaves as a Data Object project using which you can develop Data Object Services. The Project Facets page (select Properties > Project Facets on the project context menu) displays the facets associated with your current project. For more information, see page.

See also
Creating a Business Entity class on page 275
New Business Entity wizard on page 293

Creating a Business Entity class from the schema file
To create a Business Entity class from the schema of a selected source file:

1. Select an OpenEdge Data Object project and select New Business Entity from the context menu.
   The Create a Business Entity class page of New Business Entity wizard appears.

2. Specify values for the controls on the Create a Business Entity class page. See Create a Business Entity class page for detailed information about each control.

3. Click Next.
   The Select a schema file page appears.

4. The Resource name field displays the name specified in the Business Entity name field. You can enter a different name. This is an alias of Business Entity class name.

5. Select the operation as Read-only, CRUD, or CRUD and Submit. The ABL code is generated with the specified operations.

6. Select Write dataset before image, if you want the before-image data written to the file.

Note: The Write dataset before image option is selected by default if the CRUD and Submit option is selected. The before-image data is required to perform the Submit operation.
7. Select the **Select Schema from file** option.

8. In the **Schema file** field, specify a schema for the defined Business Entity class by clicking **Browse** and selecting a schema file. The schema can be either a temp-table or a dataset.

   **Note:** If you select a schema file from your current workspace, only the relative path of the project is displayed in the **Schema file** field. If the selected schema file is from outside your workspace, then the absolute path is displayed. If a schema file is not selected, the Business Entity class file is generated with a default dataset and a single temp-table.

9. The **Schema** field displays a list of all the temp-tables and datasets of the selected file; select the required temp-table or dataset.

10. Select one of the following options:

   - **Include file**: Select this option if the selected schema file is an include (.i) file. If the selected schema file is an include file, then it is included in the Business Entity class file.

      **Note:** If the Include file option is selected, the Business Entity class file is created by including the selected file. No annotations are generated for the temp-table defined in the selected file.

   - **Schema definition**: Select this option to copy the selected schema definition to a Business Entity class file. This option is enabled if the selected schema file is a procedure (.p or .w), a class (.cls), or an include (.i) file.

      **Note:** If the **Schema definition** option is selected and the selected file is an include file, the wizard adds the temp-table schema definition from the selected file to the new Business Entity class file along with annotations if the include file does not already have any annotations. In this case, the annotations are generated based on the temp-table definition. If the include file already has annotations then no additional annotations are added.

   - **Class hierarchy**: Select this option if you want to select the schema definition from class hierarchy.

11. Select **Expose as Data Object Service** to expose the defined Business Entity class as a Data Object resource.

12. Specify a Uniform Resource Identifier (URI) to access the Data Object resource in the **Resource URI** field. By default, the resource name is displayed in the field with a prefix "/". You can enter a different URI name.

13. Click **Finish**.

The wizard creates and opens the Business Entity class, which includes the temp-table definition with the Data Object interface (described in the Data Service Catalog) annotations defining the scheme attributes, in the ABL Editor.

**Note:** You can manually edit or add annotations to the Business Entity class. For more information about these annotations, see **Data Service Catalog schema properties**.

The wizard also generates a Data Object resource from the Business Entity class file. By default, the defined Business Entity class files are listed under the AppServer node in your project directory.
Note: When you add a Business Entity class file to a non-Data Object project such as OpenEdge or AppServer project, the Data Object facet gets enabled for the project. The project behaves as a Data Object project using which you can develop Data Object applications. The Project Facets page (select Properties > Project Facets on the project context menu) displays the facets associated with your current project. For more information, see page.

See also
Creating a Business Entity class on page 275
New Business Entity wizard on page 293

Using the Define Service Interface wizard

You can use the Define Service Interface wizard to add the Data Object interface annotations to your existing ABL class and procedure files. Defining the service interface involves adding Data Object interface annotations to the ABL class (.cls) and procedure (.p) files. You can then expose the Data Object annotated ABL class and external procedure files as Data Object resources.

Note: To create a Data Object resource from an ABL external procedure, the ABL external procedure must be coded to be run as a singleton procedure. For more information, see the SINGLETON option of the RUN statement in OpenEdge Development: ABL Reference guide.

To add Data Object interface annotations to an ABL class or procedure file:

1. From the Project Explorer view, select a Data Object project.
2. Right-click the Data Object project, and then select Progress OpenEdge > Define Service Interface on the context menu. The Define Service Interface wizard appears.
3. From the Definition Mode drop-down, select Data Object. By default, it is set to Data Object.
4. From the Workspace resources list, select the ABL class and procedure files to which you want to add Data Object interface annotations. This defines the Data Object interface annotations at the procedure or class file level.

Note: If the selected ABL application contains ABL routines (internal procedures, methods, and user-defined functions), they appear on the ABL routines list.

5. From the ABL routines list, select the ABL routines to which you want to add the Data Object annotations. This defines the Data Object interface annotations at the routine level.
6. Click Next. The Edit Annotation page appears.

The Edit Annotation page displays the selected resources in the Select a file list and allows you to select a schema file for a resource and associate a Data Object operation: CRUD (Create, Read, Update, and Delete), Submit, and Invoke.

7. From the Select a file list, select an ABL resource for which you want to define a schema file and associate Data Object operations: CRUD, Submit, or Invoke with ABL routines.
8. On the Main Annotations tab, specify the resource URI and schema for the selected resource. For more information, see the Main Annotations tab.

Note: Select the Enable Main Annotation check box to enable editing.
9. On the **CRUD annotations** tab, associate ABL routines with CRUD operations (**Create**, **Read**, **Update**, and **Delete**) or the **Submit** operation. You can use only one-to-one association. For more information, see the **CRUD Annotations** tab.

For example, if there is an ABL routine defined as P1 and you want to associate it with the create operation, then select P1 in the **Routine name** drop-down for the create operation. You cannot associate the ABL routine P1 with any other CRUD operation once it is associated with one operation.

To associate an ABL routine with the **Submit** operation, you must select the **Before-Image** option. The **Submit** operation is only applicable to the datasets (not temp-tables) that you select on the **Main Annotations** tab. For more information, see the **CRUD Annotations** tab.

---

**Note:** Before you associate any operation with an ABL routine, you must select a schema file for the current ABL application on the **Main annotations** tab.

---

10. On the **Invoke Annotations** tab, associate one or more ABL routines with an invoke operation. You can map as many ABL routines in an ABL file as you have available to an invoke operation. For more information, see the **Invoke Annotations** tab.

---

**Note:** You cannot associate an ABL routine that is already associated with one of the CRUD operations, with the invoke operation.

---

11. On the **Field Annotations** tab, specify the fields and appropriate field-level annotations for the selected resource. For more information, see the **Field Annotations** tab.

12. Click **Finish** to save your settings and close the **Define Service Interface** wizard.

The selected ABL class and procedure files are annotated with Data Object interface annotations, and this completes the generation of a Data Object resource for each file.

---

**Note:** When you add Data Object interface annotations for ABL resources from a non-Data Object project such as OpenEdge AppServer facets, like REST, AppServer, the Data Object facet is enabled for the project and a Data Object Service gets created. The project behaves as a Data Object project using which you can develop Data Object applications. When you add annotations to any class or procedure file, if the AppServer facet is not available for the project then the project is converted to an ABL Web App project with additional ABL REST Service and ABL Data Object Service facets and a REST based ABL Data Object service is created. The **Project Facets** page (select Properties > Project Facets on the project context menu) displays the facets associated with your current project.

---

### Creating a Data Object Service

The **New Data Object Service** wizard allows you to define a Data Object Service for a set of existing Data Object resources. Data Object resources are ABL class and procedure files with Data Object interface annotations.

---

**Note:** A default Data Object Service with the project name and Service appended to it is created when you define an OpenEdge Data Object project. For example, `<projectname>Service`. You can see the defined Data Object Service under the **Defined Services** node in your project. For more information, see Creating a Data Object project on page 274 wizard.

---

To define a Data Object Service:
1. From the **Project Explorer** view, select an OpenEdge project for which you want to create the Data Object Service.

2. Right-click the Data Object project, and then select **New > Data Object Service** on the context menu. The **New Data Object Service** wizard appears.

   **Note:** If you do not see **Data Object Service** on the context menu, then select **Other**. On the **Select a wizard** dialog, select **Progress OpenEdge > Service > Data Object Service**, and then click **Next**.

3. In the **Service name** field, specify a name for the Data Object Service.

   **Note:** You cannot have a duplicate Data Object Service name under the same OpenEdge project.

4. In the **Service relative URI** field, specify a resource Uniform Resource Identifier (URI). The default is the service name. This is relative URI for all the resources defined in the Data Object Service. The relative path must start with "/". For example, `/service_relative_URI`.

5. In the **Service description** field, enter a description for the Data Object Service. This is an optional field.

6. From the **Project** drop-down, select a project for which you want to generate the Data Object Service. By default, the field displays the project you have selected in Step 1 on page 281.

7. From the **OE Web Servers** list, select the OE Web Server instances with which you want to associate the defined Data Object Service.

8. Click **Next**. The second page of **New Data Object Service** wizard appears wherein you can select the Data Object resources. Data Object resources are ABL class and procedure files with Data Object interface annotations.

9. From the **Resources** list, select the Data Object resources. This list displays the Data Object resources of your current project.

   **Note:** Data Object resources with invalid names and names which include numeric characters, and those which include preprocessors are not supported by Data Object. If any such Data Object resource is selected, the Data Object Service will fail and display an error message.

10. Click **Select All** to add all the available Data Object resources.

    **Note:** Click **Deselect All** to clear the selection of the Data Object resources.

11. Click **Finish**.

    This creates the Data Object Service under the **Defined Services** node in the selected OpenEdge Data Object project.

    **Note:** When you define a Data Object Service for a non-Data Object project such as OpenEdge or AppServer project except ABL Web App project, the Data Object facet gets enabled for the project. The project behaves as a Data Object project enabling you to develop Data Object applications. The **Project Facets** page displays the facets associated with your current project.

**See also**

- [Data Object Service artifacts](#) on page 271
Creating an OpenEdge Data Object project on page 274
Creating a Business entity class on page 275
New Data Object Service wizard on page 301

Editing a Data Object Service

The Edit Data Object Service wizard allows you to edit a Data Object Service you have defined.

To edit a Data Object Service:

1. From the Project Explorer view, expand the Defined Services node under your Data Object project.
2. Right-click a Data Object Service, and then select Edit on the context menu. The Edit Data Object Service dialog appears.

   **Note:** You cannot modify the Service name and Project fields. These fields are disabled.

3. In the Service description field, enter a description for the Data Object Service. This is an optional field.
4. In the Service relative URI field, specify a resource Uniform Resource Identifier (URI). This is relative URI for all the resources defined in the Data Object Service. The relative path must start with "/". For example, /sampleURI.
5. Click Next. The second page of Edit Data Object Service wizard appears where you can select the Data Object resources (Data Object Service annotated ABL procedure and class files) for the defined service.
6. From the Resources list, modify your previous selection by adding or removing the resources.
7. Click Select All to add all the available resources.

   **Note:** Click Deselect All to clear the selection of the resources.

8. Click Finish to save your changes and close the wizard.

See also
Defining a Data Object Service on page 280
Deleting a Data Object Service on page 282
New Data Object Service wizard on page 301

Deleting Data Object Services

You can use the Delete option to remove defined Data Object Services from your project.

To delete Data Object Services:

1. From the Project Explorer view, expand the Defined Services node under your Data Object project.
2. Select the Data Object Services that you want to remove, and then select Delete on the context menu.
3. Click Yes to confirm the deletion when prompted.

The selected Data Object Services and their artifacts are deleted. The selected project is automatically refreshed.

See also
Defining a Data Object Service on page 280
Creating an express OpenEdge Data Object Service

OpenEdge allows you to automatically build and deploy a complete Data Object Service for a given database table, with the default configuration and components.

To create an express OpenEdge Data Object Service:

1. On the Progress Developer Studio main menu, select **File > New > OpenEdge Project**.
   
   The **New OpenEdge Project** wizard appears.

2. In the **Project name** field, enter a name for your Data Object project.

3. To change the location for the project, clear the **Use default location** check box (which is selected by default) and enter the path to the desired location (or browse to it).
   
   Selecting **Use default location** creates the root folder for the new project in your current workspace.

4. From the **Project type configuration** drop-down list, select **Data Object**.

5. Select **Express setup**.

**Note:**

To create an express Data Object application, you must meet the following prerequisites:

- **Configure servers**

  To configure servers, click **Configure servers** in the **Prerequisites** section. It opens the **OpenEdge Explorer Connections preferences** page. See OpenEdge Explorer Connections preferences for information on configuring servers.

- **Start default OE Web Server.**

  Depending upon your preference you have set up for the default server in the Preference page, Start default OE Web Server is displayed, click the link in the **Prerequisites** section.

- **Start default AppServer**

  To start the default AppServer, click **Start default AppServer** in the **Prerequisites** section section of the **New OpenEdge Project** wizard.

6. Click **Next**.

7. Specify a name for the Data Object Service in the **Data Object Service name** field.

8. From the drop-down list of the **Database connection** field, select a database connection to be associated to the Data Object project.

**Note:** If there are no database connections configured in your workspace or if you want to create a connection for another database, select **Add** to open the Add Connection Profile wizard where you can create a new database connection.
9. After you select a database connection, select a table from the drop-down list of the **Database table** field, which lists all the tables in the selected database connection.

   The wizard converts the selected database table into a temp-table definition and provides it as input for the business entity creation.

10. Click **Finish**

    The wizard then performs the following operations:

    - Builds an OpenEdge Data Object project with a database connection and REST broker.
    - Creates a business entity with the appropriate schema.
    - Creates a Data Object Service associated with the business entity.
    - Deploys the Data Object Service to the default server.

**See also**

- Creating an OpenEdge Data Object project on page 274
- Create an Express Data Object Services page on page 305

### Deploying Data Object Services

#### Starting and stopping Apache Tomcat server

The **Servers** view provides options to start and stop Apache Tomcat Java container shipped with OpenEdge.

To start Apache Tomcat:

1. Open the **Servers** view by selecting *Window > Show View > Other > Server > Servers*.
2. Right-click on an OE Web Server instance that you want to start.
3. Select **Start** on the context menu.

**Note:** The context menu options vary depending on the status of the selected OE Web Server instance. If the Web server is already started, the context menu displays the **Stop** option. To stop the Web server that is already running, select **Stop**.

Alternatively, you can start and stop Apache Tomcat using the `proenv` command prompt. For more information, see Starting and stopping Apache Tomcat. You can also start and stop Apache Tomcat and from OpenEdge Explorer and OpenEdge Management. For more information, see *OpenEdge Explorer and OpenEdge Management: Configuration* guide.

**See also**

- Publishing a Data Object service on page 286
- Creating OE Web Server server instance on page 285
Creating an OE Web Server instance

You can use the New Server wizard to define an OE Web Server instance. Note that you cannot create or define a new OE Web Server from within Progress Developer Studio for OpenEdge. To create a new OE Web Server, use OpenEdge Management.

The OE Web Server enables you to deploy Data Object Services to the Apache Tomcat Java container. You must associate Data Object Services to the OE Web Server instance to deploy them to the Tomcat. For more information, see Add and Remove dialog. The Data Object Services are deployed as WAR files on the Web server (Tomcat). A separate WAR file is created for each deployed Data Object Service.

**Note:** If you want to deploy the Data Object Services as a single WAR file, you must export them as a Data Object Web application (WAR file). For more information, see Exporting a Data Object Web Application on page 287.

To define an OE Web Server instance:

1. Access the New Server wizard by doing one of the following:
   - From the main menu, select File > New > Other > Server > Server, and then click Next.
   - Right-click on the Servers view, and click New > Server.

   **Note:** The Servers view appears by default in the OpenEdge Server perspective. You can access the Servers view by selecting Window > Show view > Other > Server > Servers from the main menu.

   The New Server wizard appears.

2. Select OE Web Server as the server type by doing one of the following:
   - Type OpenEdge OE Web Server in the Select the server type field, which filters the results to show only available OE Web Servers.
   - Select Progress Software Corporation > OE Web Server as the server type from the list provided.

   **Note:** The server host name local host is refilled in the Server's host name field. The Server name and the Server runtime environment details are also listed in the dialog.

3. If you want to specify a different server runtime environment, click Add. The New Server Runtime Environment dialog appears.

4. In the Runtime environment name field, enter the runtime environment name and click Finish.

5. To set up the runtime environment, click Configure runtime environments and follow the steps provided in Defining OpenEdge AVM runtime. To return to the Define a New Server page, click OK.

6. Click Next. The Define a new OE Web Server page appears with the Server name prefilled.

7. Do one of the following:
   - Select the name of the connection from the OpenEdge Explorer connection drop-down list.
   - Click Configure. The Progress OpenEdge>Server>OpenEdge Explorer Connections Preferences page opens. For information about how to configure an OpenEdge Explorer connection, see Adding an OpenEdge Explorer connection. To return to the Define a new OE Web Server dialog, click OK.
8. From the OE Web Server drop-down list, select an instance or click the refresh icon to verify that you are selecting from all available OE Web Servers.

   The list of OE Web Servers includes all OE Web Servers in alphabetical order, which are managed by the OpenEdge Explorer connection you specify.

   **Note:** You cannot create a new OE Web Server in Progress Developer Studio for OpenEdge; to create OE Web Server, use OpenEdge Explorer.

9. Click Next. The Add and Remove page appears showing the Data Object Services of your current workspace.

10. To configure an available Data Object Service, select service in the Available list and click Add, or double-click or service in the Available list. To configure all the available Data Object Services, click Add All.

   **Note:** You can also remove a Data Object Service from the Configured list by selecting the service and clicking Remove or double-clicking the service in the Configured list. To remove all the configured Data Object Services, click Remove All.

11. Click Finish.

   The defined OE Web Server instance appears in the Servers view. The associated Data Object Services are listed under the defined OE Web Server instance.

   **See also**
   - Starting Tomcat Web server on page 284
   - Add and remove dialog on page 304
   - Publishing a Data Object Service on page 286
   - Exporting as a Data Object Web Application on page 287

### Publishing Data Object Services

You can publish a Data Object Service(s) using the Servers view in your OpenEdge Server perspective. To publish a Data Object Service, you must create a server instance, associate the Data Object Service with the defined server, and then publish the associated Data Object Service to the Apache Tomcat Java container shipped with Progress OpenEdge.

When you install Progress Developer Studio for OpenEdge, Apache Tomcat is installed and configured with Progress OpenEdge. Before you publish to the Apache Tomcat, you must start it. For more information, see Starting and stopping Apache Tomcat server on page 284.

**Note:** The Data Object Service are published as separate WAR files when you deploy them using the Publish option from the Servers view. If you want to deploy the Data Object Services as a single WAR file, you must export them as a .WAR file. For more information, see the Exporting a Data Object Web Application on page 287.

To publish a Data Object Service:

1. From the Servers view, select the relevant server instance of the Data Object Service that you want to publish.

2. Do one of the following:
• Select Publish on the server instance’s context menu.
• Select a server instance, and then click the Publish to the server icon on the Servers view toolbar.

This publishes the selected Data Object Services to the Apache Tomcat Java container. For a published Data Object Service, a separate WAR file is generated and deployed to the Apache Tomcat Java container. For example, a Data Object Service published to the Apache Tomcat that is shipped with OpenEdge, a WAR file with the selected Data Object Service name is created in your OpenEdge installation directory, $DLC/servers/tomcat/pdsoe. A separate WAR file is created for each deployed Data Object Service.

You can view the server’s status in the Servers view beside the published Data Object Service’s node. For more information, see .

After deploying the Data Object Service, if you modify the artifacts of a project related to the deployed service, the status of the respective artifact’s module changes to Republish. You must republish the server manually if you have not set it to republish automatically so that the deployed service reflects the changes.

See also
Starting Tomcat Web server on page 284
Defining OE Web Server on page 285
Exporting as a Data Object Web Application on page 287
Data Object Service WAR file structure on page 271

Exporting a Data Object Web Application

The Data Object Web Application wizard enables you to package the Data Object Services as a Data Object Web application (WAR file) as follows:

• Data Object Service(s) into a single WAR file

Note: You can select more than one Data Object Service to package into a single Data Object Web application WAR file.

• Each Data Object Service as as a PAAR file

The WAR file that is created is used to deploy the package Data Object application to the Apache Tomcat or on any Java container that is associated with the server.

To export as a WAR file or incremental service:

1. In the Project Explorer view, select the Data Object project that contains the Data Object Services that you want to export as a Data Object Web application.
2. Right-click the project and select Export from the context menu.
   
   The Export Data Object Application dialog appears.

Note: If you do not see the Data Object Web Application option on the context menu, then select Export. On the Export wizard, select Progress OpenEdge > Data Object Web Application, and then click Next.

3. The Data Object project drop-down list displays the project you have selected in step 1. You can select a different project from the drop-down list.

Note: The Data Object project drop-down displays only the projects in your current workspace.
4. In the **Destination** drop-down list, specify a location or click **Browse** to select a different location for the Data Object Web application.

**Note:** The **Destination** drop-down displays a list of previous WAR file generation locations of your current workspace.

5. Select one of the following options:

- **Export as a WAR file** to export services by creating a single .WAR file for all the selected Data Object Services in the project.
- **Export services incrementally** to export services by creating a .PAAR file for each Data Object Service in the project.

**Note:** A dummy .PAAR file is created for each Data Object Service.

6. From the **Data Object and REST services** list, select one or more Data Object Services that you want to export.

**Note:** The **Data Object and REST services** section lists the Data Object and REST services defined in the project you have selected in the **Data Object project** drop-down list. You can also select REST services if you want to package them along with the Data Object Service as a WAR file. You can select all or deselect all services using the check box at the column header.

7. Click **Finish**.

The Data Object Web application (WAR file) or the .PAAR file is created in the location specified in the **Destination** field. By default, the WAR file is created with the project name you select in the **Data Object project** drop-down list and the .PAAR file is created with the name of the service appended to the project name.

**See also**
- Data Object Web application WAR file structure on page 271
- Publishing Data Object Services on page 286

**Using the push notification API**

The push notification API enables you to send notifications from ABL to a registered Mobile or Tablet devices. To be able to send and receive the notifications, you must enable the push notification service, see Telerik PUSH notifications.

**See also**
- Push notification API reference on page 310

**Connecting to a push notification server**

Before sending notifications, you must connect to the push notification server using an instance of the `OpenEdge.Mobile.PushNotificationService` class. This `PushNotificationService` requires a URI (representing the server location) and an API key (required to identify the notification sender).
The example below illustrates creating a connection to the push notification server on Telerik server.

**Note:** Depending on your application design, you may need only a single connection for the duration of your ABL session; a single PushNotificationService instance can send multiple notification messages.

```abl
block-level on error undo, throw.
using OpenEdge.Net.URI.

define variable oURI as URI no-undo.
define variable cApiKey as character no-undo.
define variable oPushService as PushNotificationService no-undo.

assign oURI = NEW URI('http', 'api.everlive.com/v1').
cApiKey = "<API key of Telerik backend service>"
oPushService = PushNotificationService:Instance(oURI, cApiKey).
```

A new URI object is created using a constructor, providing at least the scheme and host. The other elements of the URI can be added using properties (most) or methods (queries).

The push notification service only uses the scheme, host, and port of the provided URI; the other parts are discarded. A URI can additionally be derived from a string using the static Parse() method as shown below.

```abl
using OpenEdge.Net.URI.

define variable oURI as URI no-undo.
oURI = URI:Parse('http://api.everlive.com/v1').
```

See also
Push notification API reference on page 310

**Sending and scheduling notification messages**

The push notification service provides the following methods for sending and scheduling notification messages:

- **SendNotification()**: sends a notification message immediately
- **ScheduleNotification()**: sends a message at a specified time in the future

The SendNotification() and ScheduleNotification() methods each contain a message text and a badge count (only valid for iOS), and return a NotificationServiceError in the event of an error. They send notifications to all registered devices.
The below example builds on the example in Connecting to a push notification server on page 288 and illustrates the SendNotification() and ScheduleNotification() methods.

```plaintext
define variable cMessage as character no-undo.
define variable iBadgeCount as integer no-undo.
define variable tSendAt as datetime-tz no-undo.
assign cMessage = 'App users should know this fact'.
/* send immediately */
oPushService:SendNotification(cMessage, iBadgeCount).

/* send in an hour's time */
assign cMessage = 'App users should know this fact, but not right now'
tSendAt = add-interval(now, 1, 'hours').
oPushService:ScheduleNotification(
    cMessage,
    iBadgeCount,
    tSendAt,
    true). /* use device time zone? */
catch eNSE as NotificationServiceError:
    message
        string(eNSE:GetErrorMessage())
    view-as alert-box title 'NotificationServiceError'.
end catch.
```

The OpenEdge.Mobile.TelerikPushNotificationAdmin API also enables you to modify or delete notifications, send normal push notifications.

See also

Push notification API reference on page 310

Creating notifications with device filtering criteria

The OpenEdge.Mobile.PushNotificationMessageBuilder class enables you to create customized notifications. You can create notifications with a device filtering criteria and send messages only to certain devices. You can use various filtering criteria like physical device, user, or geographic location.

You can use the OpenEdge.Mobile.PushNotificationMessageBuilder class to perform the following operations:

1. Call the Send() or Schedule() method
2. Add the filtering criteria
3. Retrieve the completed payload using the Payload property.
The example below illustrates the `SendNotification()` and `ScheduleNotification()` methods using the `OpenEdge.Mobile.PushNotificationMessageBuilder` class.

```plaintext
define variable oPayload as JsonObject no-undo.
/* Message builder equivalent of SendNotification() method */
assign oPayload = PushNotificationMessageBuilder
    :Send(cMessage, iBadgeCount)
    :Payload.

/* Message builder equivalent of ScheduleNotification() method */
assign oPayload = PushNotificationMessageBuilder
    :Schedule(cMessage, iBadgeCount)
    :At(add-interval(now, 1, 'hour'))
    :Payload.

/* Perform the send */
oPushService:SendNotification(oPayload).
```

The push notification server, by default, provides support for the following basic filters:

<table>
<thead>
<tr>
<th>Filter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Devices</td>
<td>Enables you to choose specific devices to which the notifications are sent using the <code>OnDevice</code>, <code>NotOnDevice</code>, <code>IncludeDevice</code>, and <code>ExcludeDevice</code> methods. These methods filter the devices by device ID (character).</td>
</tr>
<tr>
<td>Channels</td>
<td>Classifies devices by channel using the <code>IncludeChannels</code>, <code>ExcludeChannels</code>, and <code>MatchChannels</code> methods. These methods use a <code>JsonArray</code> argument. Note: <code>Channels</code> is a predefined array column.</td>
</tr>
</tbody>
</table>

Note: For detailed information about other types of filtering capabilities provided, see Telerik PUSH notifications.
The below example illustrates sending messages to the three specified devices.

```plaintext
/* send to specific devices */
define variable cDeviceId as character extent 3 no-undo.
assign cDeviceId[1] = '<first-device-id>'
cDeviceId[2] = '<second-device-id>'
cDeviceId[3] = '<third-device-id>'.
assign oPayload = PushNotificationMessageBuilder
  :Send(cMessage, iBadgeCount)
  :IncludeDevice(cDeviceId[1])
  :IncludeDevice(cDeviceId[2])
  :IncludeDevice(cDeviceId[3])
  :Payload.
/* Perform the send */
oPushService:SendNotification(oPayload).
```

The `OpenEdge.Mobile.PushNotificationMessageBuilder` class provides API for adding filter criteria on any column. The supported data types for the columns are:

- Array
- Date
- Geo
- Logical
- Numeric
- Object
- Pointer
- String

Each data type has its own `Add<type>Filter()` method, which uses the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter name</td>
<td>A character expression that matches the column name.</td>
</tr>
<tr>
<td>Filter value</td>
<td>A value based on which the devices are filtered.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The type of data required depends on the filter type. For instance, an Array filter requires an instance of a JsonArray object; a String filter requires a character value.</td>
</tr>
<tr>
<td>Operand (such as Equals or Includes)</td>
<td>Operands represented by the <code>OpenEdge.Mobile.PushNotificationFilterOperandEnum</code> enumeration.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Different data types support different operands. For instance, a Geo filter supports Near, NearSphere, and Within but not Equals or Includes.</td>
</tr>
</tbody>
</table>
See also
Push notification API reference on page 310

Reference

New Business Entity wizard

The **New Business Entity** wizard allows you to create an ABL class (.cls) which automatically includes the Data Object interface annotations. You can then expose the Data Object interface annotated Business entity class as a Data Object resource.

Alternatively, you can create an ABL class file using the **New ABL Class** wizard, and use the **Define Service Interface** wizard to add Data Object interface annotations to expose the defined class file as a Data Object resource.

You can access the **New Business Entity** wizard by selecting *File > New Business Entity* in the OpenEdge Server perspective.

---

**Note:** If you do not see **Business Entity** on *File > New*, then select *File > New > Others*. On the wizard selection page, select > **Progress OpenEdge > Business Components > Business Entity**, to open the **New Business Entity** wizard.

The **New Business Entity** wizard includes the two pages, described in the following topics.

---

Create a Business entity class page

The **Create a Business entity class** page of the **New Business Entity** wizard allows you to define a Business entity class with pre-defined Data Object interface annotations added to it.

This page includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Package root</strong></td>
<td>Specifies a currently open project in which you want to the Business entity class code and other project code. Click <strong>Browse</strong> if you want to select a project other than the current one (the default project).</td>
</tr>
<tr>
<td><strong>Package</strong></td>
<td>(Optional) Specifies a package name, corresponding to a sub-folder of the package root, to contain the Business entity class file. Click <strong>Browse</strong> or type the folder path. A period (.) must separate each subfolder from its parent folder in the path name.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| **Business entity name** | (Required) Specifies the name of the Business entity class. The name must begin with a letter; spaces and most non-alphanumeric characters are not allowed. The .cls extension will be appended automatically.  
  **Note:** You cannot specify duplicate class names under the same project root. |
| **Final Modifier** | Select this check box to specify that inheritance from this Business entity class is disallowed. A FINAL keyword is included in the generated ABL code.  
  **Note:** A final class cannot be abstract. |
| **Abstract Modifier** | Specifies that the Business entity class is abstract and cannot be instantiated. An abstract class is designed to serve as a super class from which other classes inherit and implement members. Therefore, an abstract class cannot be final.  
  **Note:** If you select the Abstract check box, the Expose as Data Object Service check box on the next page of the wizard will be disabled. |
| **Widget Pool Modifier** | If selected, specifies that a USE-WIDGET-POOL keyword is included in the generated ABL code. |
| **Serializable Modifier** | If selected, marks the class as serializable by including the SERIALIZABLE modifier in the generated ABL code.  
  **Note:** If the class inherits from a super class that is not serializable, then it cannot be marked as serializable. |
| **Inherits**       | Displays another Business entity class in the current project as a super class from which the new class inherits state and behavior. Click **Browse** to select a different class path or enter the super class's name. A period (.) must separate each node from its parent folder in the path name.  
  **Note:** The super class cannot be generic. |
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implements</td>
<td>(Optional) Specifies one or more interfaces in the current project that the class implements. Click Add and select the desired interfaces at the Interface Selection dialog. Use Remove to remove an interface from those added to the list.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The class cannot implement a generic interface.</td>
</tr>
<tr>
<td>Default constructor</td>
<td>Specifies that the Business entity class is to include a default constructor method.</td>
</tr>
<tr>
<td>Destructor</td>
<td>Specifies that the Business entity class is to include a destructor method.</td>
</tr>
<tr>
<td>Super class constructors</td>
<td>Select this check box to specify that the Business entity class is to include all constructor methods declared in the parent class.</td>
</tr>
<tr>
<td>Error-handling statement</td>
<td>Adds one of the following error-handling statements to the generated ABL code:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Block level:</strong> Includes the BLOCK-LEVEL ON ERROR UNDO, THROW statement in the generated ABL code</td>
</tr>
<tr>
<td></td>
<td>• <strong>Routine level:</strong> Includes the ROUTINE-LEVEL ON ERROR UNDO, THROW statement in the generated ABL code</td>
</tr>
<tr>
<td>Throw a Not Implemented exception</td>
<td>If selected, specifies that &quot;METHOD NOT IMPLEMENTED&quot; error code is added to stubs for members implemented via interface and stubs for inherited abstract members.</td>
</tr>
<tr>
<td>Return a default value</td>
<td>If selected, specifies that default return values are added to stubs for members implemented via interface and stubs for inherited abstract members.</td>
</tr>
<tr>
<td>Description</td>
<td>(Optional) Specifies a description for the Business entity class. This text will appear in the file header.</td>
</tr>
<tr>
<td>Purpose</td>
<td>(Optional) Specifies an explanation of the purpose of the Business entity class. This text will appear in the file header.</td>
</tr>
<tr>
<td>Next</td>
<td>Takes you to the next page of <strong>New Business Entity</strong> wizard.</td>
</tr>
</tbody>
</table>

**See also**

[Select a schema file page](#) on page 295

**Select a schema file page**

The **Select a schema file** page of the **New Business Entity** wizard allows you to define schema and Data Object operations that you want to generate for a Business entity class file. You can expose the Data Object interface annotated Business entity class file as a Data Object resource.
This page includes the following controls:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource Name</td>
<td>Displays the Business entity name specified in the <strong>Business entity name</strong> field. You can enter a different name. This is an alias of the Business entity class name.</td>
</tr>
<tr>
<td>Read-Only operations</td>
<td>Specifies the access method as a read-only operation. The ABL code (including the Data Object interface annotations) is generated with Read method.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> When you select dataset with multiple relations, <strong>Invoke</strong> methods are also generated for each temp-table.</td>
</tr>
<tr>
<td>CRUD operations</td>
<td>Specifies the access method for the operation as create, read, update, or delete. The ABL code (including the Data Object interface annotations) is generated with predefined CRUD (create, read, update, and delete) operations.</td>
</tr>
<tr>
<td>CRUD and Submit operations</td>
<td>Specifies the access method for the operation as Create, Read, Update, Delete, or Submit.</td>
</tr>
<tr>
<td>Write dataset before image</td>
<td>Writes the before-image data to the new Business entity class file.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The <strong>Write dataset before image</strong> option is selected by default if the <strong>CRUD and Submit</strong> option is selected. The before-image data is required to perform the submit operation.</td>
</tr>
<tr>
<td>Select database table</td>
<td>Creates a Business entity class from the selected database table.</td>
</tr>
<tr>
<td>Connection</td>
<td>(Required if <strong>Select database table</strong> is selected) Specifies the database connection that you want to use to dynamically define the temp table.</td>
</tr>
<tr>
<td>Table</td>
<td>(Required if <strong>Select database table</strong> is selected) Specify the database table from selected database connection.</td>
</tr>
<tr>
<td>Select schema from file</td>
<td>Creates a Business entity class from the selected schema file.</td>
</tr>
<tr>
<td>Control</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Schema file</strong></td>
<td><em>(Required if Select schema from file is selected)</em> Specifies a schema for the defined Business entity class. The schema can be either a temp-table or dataset. Click <strong>Browse</strong> to select a schema file.</td>
</tr>
<tr>
<td></td>
<td>If you select schema file, which is from your current workspace, only the relative path of the project is displayed in the <strong>Schema file</strong> field. If the selected schema file is from outside your workspace, then the absolute path is displayed.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>If a schema file is not selected, the Business entity class file is generated with a default dataset and a single temp-table.</td>
</tr>
<tr>
<td><strong>Schema</strong></td>
<td><em>(Required if Select schema from file is selected)</em> Displays a list of all the temp-tables and datasets of the selected file.</td>
</tr>
<tr>
<td><strong>Include file</strong></td>
<td>Is enabled if the selected schema file is an include (.i) file.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>If the selected schema file is an include file, then it is included in the Business entity class file.</td>
</tr>
<tr>
<td><strong>Schema definition</strong></td>
<td>Copies the selected schema definition to a Business entity class file. This is the default option.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>This option is enabled if the selected schema file is a procedure (.p or .w), a class (.cls), or an include (.i) file.</td>
</tr>
<tr>
<td><strong>Class Hierarchy</strong></td>
<td>Enables you to select the schema definition from class hierarchy.</td>
</tr>
<tr>
<td><strong>Expose as Data Object Service</strong></td>
<td>Adds Data Object interface annotations to the defined Business entity class file and exposes it as a Data Object resource.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>If the <strong>Expose as Data Object Service</strong> check box is clear, the Business entity class is generated without the Data Object interface annotations. The defined Business entity class file cannot be exposed as a Data Object resource. Alternatively, you can use the <strong>Define Service Interface</strong> wizard to add Data Object interface annotations to a Business entity class file.</td>
</tr>
<tr>
<td><strong>Finish</strong></td>
<td>Completes the wizard and opens the defined Business entity class including the Data Object interface annotations in the ABL Editor. This also completes the generation of Data Object resource from the Business entity class file.</td>
</tr>
</tbody>
</table>
See also
New Business Entity Class wizard on page 293
Using the Define Service Interface wizard on page 279

Define Service Interface wizard

Define Service Interface page
The Define Service Interface page allows you to add the Data Object interface annotations to your existing ABL class and procedure files.
Access this wizard by selecting Progress OpenEdge > Define Service Interface on the project context menu.
This page includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition mode</td>
<td>Specify the interface definition type as Data Object from the drop-down.</td>
</tr>
<tr>
<td>Workspace resources</td>
<td>Displays a list of ABL class and procedure files of your current workspace.</td>
</tr>
<tr>
<td>ABL routines</td>
<td>Displays a list of defined internal procedures, methods, and user-defined functions (UDFs).</td>
</tr>
<tr>
<td>Backup selected resources</td>
<td>Select this check box to back up the selected resources before adding the Data Object interface annotations.</td>
</tr>
<tr>
<td>Location</td>
<td>Specify a location where you want to save the back-up resources. Click Browse to specify a back-up location.</td>
</tr>
<tr>
<td>Next</td>
<td>Takes you to the Edit Annotations page of the Define Service Interface wizard.</td>
</tr>
</tbody>
</table>

Note: You must select at least one ABL routine to have the Next button enabled.

See also
Defining Data Object Service annotations on page 279
Edit Annotations page on page 298

Edit Annotation page
The Edit Annotations page of the Define Service Interface wizard allows you to select a schema file for the selected ABL class or procedure file, and associate a Data Object operation with an ABL routine.
This page includes the following options:
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select a file</td>
<td>Displays a list of ABL classes and procedures selected in the Workspace resources list on the first page of the Define Service Interface wizard.</td>
</tr>
<tr>
<td>Main annotation</td>
<td></td>
</tr>
<tr>
<td>Enable Main Annotation</td>
<td>Select to enable editing the Main Annotations page.</td>
</tr>
<tr>
<td>Execution mode</td>
<td>Specify singleton as the execution mode from the drop-down. This is the only default execution mode for both the procedure and class files.</td>
</tr>
<tr>
<td>Return value</td>
<td>Select this check box if the selected procedure returns a string value. A useReturnValue parameter is added with the Data Object interface annotations to the procedure file. By default, this option is not selected.</td>
</tr>
<tr>
<td>Note:</td>
<td>The Return value check box is disabled for a class file.</td>
</tr>
<tr>
<td>Before-image</td>
<td>Sets the writeDatasetbeforeImage parameter value to TRUE for the selected ABL routine. By default, this option is disabled for the Data Object resource.</td>
</tr>
<tr>
<td>Resource name</td>
<td>Displays the name of the resource selected in the Select a file list. You can specify a different name for the resource.</td>
</tr>
<tr>
<td>Resource URI</td>
<td>Specify a Uniform Resource Identifier (URI) to access a resource. By default, the resource name is displayed in the field with a prefix &quot;/&quot;. You can enter a different URI name.</td>
</tr>
<tr>
<td>Schema file</td>
<td>Specify a schema for the selected resource. The schema can either be a temp-table or dataset. Click Browse to select a schema file.</td>
</tr>
<tr>
<td>Note:</td>
<td>You can select the schema file either from the workspace or file system. If the selected schema file is from your current workspace, only the relative path of the project is displayed in the Schema file field. If the selected schema file is from outside your workspace, then the absolute path is displayed.</td>
</tr>
<tr>
<td>Schema</td>
<td>Displays a list of the temp-tables and datasets of the schema selected in the Schema file field.</td>
</tr>
<tr>
<td>Schema name</td>
<td>Displays the schema definition selected from the Schema list.</td>
</tr>
<tr>
<td>Main annotation for the above selected file</td>
<td>Displays the file-level Data Object interface annotations defined for the selected ABL class or procedure file.</td>
</tr>
<tr>
<td>CRUD annotations</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Operation</td>
<td>Displays the CRUD (Create, Read, Update, and Delete) and Submit operations that you can associate with ABL routines.</td>
</tr>
<tr>
<td></td>
<td>You can have only one-to-one association. For example, if there is an ABL routine defined as P1 and which you want to associate with a create operation, then select P1 in the Routine name drop-down for the create operation. You cannot associate the ABL routine P1 with any other CRUD operation when it is associated with one operation.</td>
</tr>
<tr>
<td></td>
<td>To associate an ABL routine with the Submit operation, you must select the Before-Image option.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The Submit operation is only applicable to the datasets (not temp-tables) that you select in the Main Annotation tab.</td>
</tr>
<tr>
<td>Routine name</td>
<td>Displays a list of ABL routines of the selected ABL procedure or class file.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This list displays only the ABL routines selected in the ABL routines list on the first page of the Define Service Interface wizard.</td>
</tr>
<tr>
<td>Alias</td>
<td>(Optional) Specify an alias for an operation.</td>
</tr>
<tr>
<td>Return value</td>
<td>(Optional) Select this check box if the ABL routine selected in the Routine name field returns a string value. By default, this option is not selected.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The Return value check box is enabled only for internal procedures. This is disabled for both the user-defined functions (UDFs) and methods.</td>
</tr>
<tr>
<td>Before-Image</td>
<td>Sets the writeDatasetbeforeImage parameter value to TRUE for the selected ABL routine.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This option only applies to datasets and not to temp-tables.</td>
</tr>
<tr>
<td>Data Object detail annotation</td>
<td>Displays the ABL routine-level Data Object interface annotations.</td>
</tr>
<tr>
<td>Invoke annotations</td>
<td></td>
</tr>
<tr>
<td>Operation</td>
<td>Displays invoke or count operation that you can associate with an ABL routine.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> You cannot associate an ABL routine with an invoke or count operation, which is already associated with one of the CRUD operations.</td>
</tr>
<tr>
<td></td>
<td>You can not associate more than one count operation for the selected resource.</td>
</tr>
</tbody>
</table>
### Option | Description
--- | ---
**Routine name** | Displays a list of ABL routines of the selected ABL procedure or class file.  
*Note:* This list displays the ABL routines selected in the **ABL routines** list on the first page of the **Define Service Interface** wizard.

**Alias** | *(Optional)* Specify an alias for an operation.

**Return value** | *(Optional)* Select this check box if the ABL routine selected in the **Routine name** field returns a string value. By default, this option is not selected.  
*Note:* The **Return value** check box is enabled only for internal procedures. This is disabled for both the UDFs and methods.

**Before-Image** | Sets the `writeDatasetbeforeImage` parameter value to **TRUE** for the selected ABL routine.

**Data Object detail annotation** | Displays the routine-level Data Object interface annotations.

**Field annotations**

| **TempTable** | Displays the TempTables available for the class file. Select the required TempTable. |
| **Field** | Select the field that you want to add annotations for. |
| **Annotation** | Select the type of annotation that you want to add. |

**Finish** | The Data Object interface annotations are added to the selected ABL applications and closes the **Define Service Interface** wizard.  
*Note:* When you add a Data Object interface annotations for ABL resources from a non-Data Object project, such as OpenEdge or AppServer project, the Data Object facet is enabled for the project and a Data Object service is created. The project behaves as a Data Object project, enabling you to develop Data Object Services. When you add annotations to any class or procedure file, if the AppServer facet is not available for the project, then the project is converted to an ABL Web App project with additional ABL REST Service and ABL Data Object Service facets, and a REST based ABL Data Object service is created. The **Project Facets** page (select **Properties > Project Facets** on the project context menu) displays the facets associated with your current project.

---

**New Data Object Service wizard**

The **New Data Object Service** wizard allows you to create a Data Object Service for a set of existing Data Object resources. Access this wizard by choosing **New > Data Object Service** on the Data Object project context menu. The **New Data Object Service** wizard includes the following options:
### Option | Description
--- | ---
**Service name** | Specify a name for the Data Object Service. This is a mandatory field.
**Service description** | Specify a description for the Data Object Service. This is an optional field.
**Service relative URI** | Specify a Uniform Resource Identifier (URI) for a Data Object Service. For example, `/relative URI`.  
*Note:* The relative URI must start with `"/sampleURI"`.
**Project** | Select a project wherein you want to generate all the Data Object Service related files (artifacts).  
*Note:* By default, the project name you select when creating the Data Object Service is displayed in the **Project** field.
**OE Web Servers** | Select the OE Web Server instances with which you want to associate a defined Data Object Service.  
*Note:* The list displays the OE Web Server instances registered with OpenEdge Management.
**Resources** | Displays a list of Data Object resources. These are Data Object interface annotated ABL class and procedure files.
**Select All** | Selects all the Data Object resources available on the **ABL routines** list.
**Deselect All** | Clears the selection of all Data Object resources on the **ABL routines** list.

### Note: When you create a Data Object Service for a non-Data Object project such as OpenEdge or AppServer project, the Data Object facet gets enabled for the project. The project behaves as a Data Object project enabling you to develop Data Object applications. The **Project Facets** page (select **Properties > Project Facets** on the project context menu) displays the facets associated with your current project. For more information, see page.

### See also
- Creating a Data Object Service on page 280

### Data Object Service context menu
Right-clicking on a Data Object Service displays the Data Object Service context menu.
Note: The defined Data Object Services are available under the Defined Services node in the Data Object project.

The Data Object Service context menu has the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit</td>
<td>Opens the Editing a Data Object Service on page 282 dialog which allows you edit a defined Data Object Service.</td>
</tr>
<tr>
<td>Delete</td>
<td>Removes the selected defined Data Object Services (along with their REST and Data Object artifacts) from the project.</td>
</tr>
<tr>
<td>Export</td>
<td>Packages a Data Object service(s), or REST service(s) as separate Data Object Web applications each or into one single Data Object Web application (WAR file) or to export services incrementally by creating one .PAAR file for each REST or Data Object service in the project.</td>
</tr>
</tbody>
</table>

See also
Defining a Data Object Service on page 280
Editing a Data Object Service on page 282
Deleting a Data Object Service on page 282

Export Data Object Web Application wizard

The Export Data Object Web Application wizard enables you to package a Data Object Service(s), or REST service(s) as separate Data Object Web applications into one single Data Object Web application (WAR file) or to export services incrementally by creating one .PAAR file for each REST or Data Object Service in the project.

Access this dialog box by selecting Export > Data Object Web Application on the Data Object project context menu.

The dialog box includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Object project</td>
<td>Specifies a Data Object project that contains the Data Object Services that you want to export as a Data Object web application.</td>
</tr>
<tr>
<td>Destination</td>
<td>Allows you to select a location from the drop-down list. Click Browse to select a different location for the file.</td>
</tr>
</tbody>
</table>

Note: The Destination drop-down menu displays a list of previous WAR or .PAAR file generation locations of your current workspace.

Server type | Select OE Web Server or Progress Application Server for OpenEdge to export a Data Object Service or a REST service. |
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Export as a WAR file</td>
<td>Select to export services by creating a single .WAR file for all the selected REST or Data Object Services in the project.</td>
</tr>
<tr>
<td>Export services incrementally</td>
<td>Select to export each service separately by creating a .PAAR file for each selected REST or Data Object Service in the project.</td>
</tr>
<tr>
<td>Data Object and REST services</td>
<td>Allows you to select Data Object and REST services from the list. Note: The Data Object and REST services section lists the Data Object and REST services defined in the project selected in the Data Object project drop-down.</td>
</tr>
</tbody>
</table>

See also
Exporting a Data Object Web application on page 287

Add and Remove dialog

The Add and Remove dialog enables you to associate or disassociate Data Object Services with an OE Web Server instance.

Access this dialog by selecting an OE Web Server instance in the Servers view, and then choose Add and Remove on the context menu.

Note: You can also access the Add and Remove page from the New Server wizard. For more information, see the Creating an OE Web Server instance on page 285.

This page includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available</td>
<td>Displays a list of available Data Object Services that you can associate with an OE Web Server instance.</td>
</tr>
<tr>
<td>Configured</td>
<td>Displays a list of Data Object Services associated with an OE Web Server instance.</td>
</tr>
<tr>
<td>Add</td>
<td>Moves a selected Data Object Service to the Configured list.</td>
</tr>
<tr>
<td>Add All</td>
<td>Moves all available Data Object Services to the Configured list.</td>
</tr>
<tr>
<td>Remove</td>
<td>Moves a selected Data Object Service to the Available list.</td>
</tr>
<tr>
<td>Remove All</td>
<td>Moves all the associated Data Object Services to the Available list.</td>
</tr>
<tr>
<td>Finish</td>
<td>Finish adding and removing Data Object Services.</td>
</tr>
</tbody>
</table>
Create an Express Data Object application page

This page appears when you select the **Express setup** option on the **New OpenEdge Project** wizard. The **Create an Express Data Object application** page allows you to automatically build and deploy a complete Data Object Service for a given database table with the default configuration and components.

The following controls are available on this page:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Object Service name</td>
<td>Specifies the name of the Data Object Service.</td>
</tr>
<tr>
<td>Database connection</td>
<td>Displays a list of database connections configured for the current workspace.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> If there are no database connections configured in your workspace or if you want to create a connection for another database, select <strong>Add</strong> to open the wizard where you can create a new database connection.</td>
</tr>
<tr>
<td>Database table</td>
<td>Displays a list of tables available in the selected database connection. The selected database table is converted to a temp-table definition and provided as input for business entity creation.</td>
</tr>
</tbody>
</table>

Data Service Catalog schema properties

The JSDO provides a set of built-in methods that correspond to the Business Entity CRUD and Submit operations. For these operations to work successfully together, the data (and its schema) are shared between the operations. The Data Service Catalog file includes an entry for each built-in method and each invocation (or Invoke) method for the Data Object resource. It also includes the schema for the resource. The JSDO uses the schema information for its built-in CRUD and Submit methods. For each method, the catalog provides parameter mapping information. This information (metadata) from the catalog is used to construct the HTTP request for the operation calls made to the REST adapter.
OpenEdge provides the following Data Service Catalog schema properties:

- `foreignkey`
- `method`
- `primarykey`
- `readOnly` (can only be added using the Field property annotation)
- `recordName` (can only be added using the Entity property annotation)
- `required`

The entity level properties can be added using the Entity property annotation. See Entity property annotation for more information.

The field level properties can be added using the Field property annotation. See Field property annotation for more information.

The following table provides a summary of the Data Service Catalog schema properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Parent node</th>
<th>Type</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>foreignkey</td>
<td>Entity</td>
<td>Object</td>
<td>undefined</td>
</tr>
<tr>
<td>method</td>
<td>Operation</td>
<td>String</td>
<td>undefined</td>
</tr>
<tr>
<td>primarykey</td>
<td>Entity</td>
<td>Array of strings</td>
<td>undefined</td>
</tr>
<tr>
<td>readOnly</td>
<td>Field of Entity</td>
<td>Boolean</td>
<td>false</td>
</tr>
<tr>
<td>recordName</td>
<td>Entity</td>
<td>String</td>
<td>undefined</td>
</tr>
<tr>
<td>required</td>
<td>Field of Entity</td>
<td>Boolean</td>
<td>false</td>
</tr>
</tbody>
</table>

**foreignkey property**

The `foreignkey` property, `@openapi.openedge.entity.foreignkey`, is added at the entity level in the schema to specify the foreign key of an entity. It specifies the relationships of an entity to other entities, which may not be defined in the same resource (for a DataSet) or in the same catalog.

Progress Developer Studio for OpenEdge generates annotations to specify the foreign keys if it can find fields that correspond to the indexes in a unique index on another table.

When you generate a Business Entity class file from an express Data Object project or from a database table (that has a foreign key constraint on it) in the New Business Entity wizard, Progress Developer Studio for OpenEdge reads the respective information from the database and adds the foreignkey annotation above the temp-table definition.
Sample foreignkey annotation for a table that has a foreign key constraint on it:

```plaintext
@openapi.openedge.entity.foreignkey (fields="ORDENUM,Linenum", parent="ORDER", parentFields="ORDENUM,Linenum").

DEFINE TEMP-TABLE ttOrderLine BEFORE-TABLE bttOrderLine
FIELD Discount AS INTEGER INITIAL "0" LABEL "Discount"
FIELD ExtendedPrice AS DECIMAL INITIAL "0" LABEL "Extended Price"
FIELD Itemnum AS INTEGER INITIAL "0" LABEL "Item Num"
FIELD Linenum AS INTEGER INITIAL "0" LABEL "Line Num"
FIELD OrderLineStatus AS CHARACTER INITIAL "Ordered" LABEL "Order Line Status"
FIELD Ordernum AS INTEGER INITIAL "0" LABEL "Order Num"
FIELD Price AS DECIMAL INITIAL "0" LABEL "Price"
FIELD Qty AS INTEGER INITIAL "0" LABEL "Qty"
INDEX itemnum Itemnum DESCENDING
INDEX orderline IS PRIMARY UNIQUE Ordernum DESCENDING Linenum DESCENDING
INDEX OrderLineStatus OrderLineStatus DESCENDING
INDEX SYS_001_000000004 Ordernum DESCENDING.

DEFINE DATASET dsOrderLine FOR ttOrderLine.
```

You can manually change the value of the foreignkey annotation in the Business Entity class file.

See also
DataService Catalog schema properties on page 305

method property

The method property, `@openapi.openedge.method.property`, is added at the operation level in the schema to specify additional properties for CRUD and submit methods.

Note: The method property annotation must be added manually to the Business Entity class file.

Sample method property annotation:

```plaintext
@openapi.openedge.method.property (name="capabilities", value="top, skip, id, orderBy").
```

See also
DataService Catalog schema properties on page 305

primarykey property

The primarykey property, `@openapi.openedge.entity.primarykey`, is added at the entity level in the schema to specify the primary key of an entity.

Progress Developer Studio for OpenEdge generates an annotation for the primary key if the primary index is unique.

When you generate a Business Entity class file from an express Data Object project or from a database table in the New Business Entity wizard, Progress Developer Studio for OpenEdge reads the database for information about the primary keys defined on the table and accordingly adds the primarykey annotation above the temp-table definition.
Sample primarykey annotation:

```java
@openapi.openedge.entity.primarykey (fields="EmpNum").

DEFINE TEMP-TABLE ttBenefits BEFORE-TABLE bttBenefits
FIELD EmpNum AS INTEGER INITIAL "0" LABEL "Emp No"
INDEX EmpNo IS PRIMARY UNIQUE EmpNum DESCENDING.
```

You can manually change the value of the primarykey annotation in the Business Entity class file.

See also
DataServiceCatalog schema properties on page 305

required property

The required property, `@openapi.openedge.entity.required`, specifies the mandatory fields of an entity. It indicates whether a field can have a null value (unknown value in ABL or undefined in JSON).

In ABL, there is no temp-table support to specify whether or not a field is mandatory. So, the annotations for required fields are obtained from the database table definition when you generate a Business Entity class file from an express Data Object project or from a database table in the New Business Entity wizard.

Sample required annotation:

```java
@openapi.openedge.entity.required (fields="HealthCare,MedicalSpending").

DEFINE TEMP-TABLE ttBenefits BEFORE-TABLE bttBenefits
FIELD HealthCare AS CHARACTER LABEL "Health Care"
FIELD MedicalSpending AS INTEGER INITIAL "0" LABEL "Medical Spending"
FIELD StockPurchase AS INTEGER INITIAL "0" LABEL "Stock Purchase"
FIELD XTZDEPENDENTCARE AS INTEGER INITIAL "0" LABEL "Dependent Care"
INDEX EmpNo IS PRIMARY UNIQUE EmpNum DESCENDING
```

The required annotation is added at the entity level to avoid a large number of annotations being added to a temp-table. But, when generating the Data Service Catalog, the required property is added at the field level.

See also
DataServiceCatalog schema properties on page 305

Entity property

The Entity property, `@openapi.openedge.entity.property`, uses the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Specifies the name of the property to be added at the entity level.</td>
</tr>
</tbody>
</table>
### Entity property

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Specifies the value of the property as a string. The value can also be a comma delimited list of strings.</td>
</tr>
</tbody>
</table>
| Type      | (Optional) Indicates how the property is generated in the Data Service Catalog file. The type can be a string, integer, boolean, or an array.  

**Note:** The default type is a string. If the type is an array, the generated code looks like a JSON array of strings where each item corresponds to each item in the comma delimited value string. |

Sample `Entity` property annotation:

```java
@openapi.openedge.entity.property (name="recordName", value="Name", type="string").
```

### See also

Data Object Catalog schema properties on page 305

### Field property

The Field property, `@openapi.openedge.entity.field.property`, uses the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>Specifies the name of the field in the schema.</td>
</tr>
<tr>
<td>Name</td>
<td>Specifies the name of the property to be added at the field level.</td>
</tr>
<tr>
<td>Value</td>
<td>Specifies the value of the property as a string. The value can also be a comma delimited list of strings.</td>
</tr>
</tbody>
</table>
| Type      | (Optional) Indicates how the property is generated in the Data Service Catalog file. The type can be a string, integer, boolean, or an array.  

**Note:** The default type is a string. If the type is an array, the generated code looks like a JSON array of strings where each item corresponds to each item in the comma delimited value string. |
Sample Field property annotation:

```java
@openapi.openedge.entity.field.property (field="numOrders", name="readOnly", value="true", type="boolean").
```

See also
Data Service Catalog schema properties on page 305

## Push notification API reference

The ABL push notification service primarily uses the following API classes:

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenEdge.Mobile.PushNotificationService</td>
<td>Sends push notifications to devices.</td>
</tr>
<tr>
<td>OpenEdge.Mobile.ApplicationPlatformEnum</td>
<td>Describes the supported Mobile platforms for push notifications.</td>
</tr>
<tr>
<td>OpenEdge.Net.URI</td>
<td>Represents a network address, which represents the push notification Service.</td>
</tr>
<tr>
<td>OpenEdge.Net.UriSchemeEnum</td>
<td>Contains members for HTTP and HTTPS.</td>
</tr>
</tbody>
</table>

**Note:** The OpenEdge.Net procedure libraries containing the above classes are not a part of the standard libraries and must be added to the project’s PROPATH manually. The OpenEdge.Net.pl file is located at `@{DLC}\tty\netlib` or `@{DLC}\gui\netlib` (based on the runtime), where DLC is the default installation directory.

See also
Using the push notification API on page 288

**OpenEdge.Mobile.TelerikPushNotificationAdmin API**

You can also use the `OpenEdge.Mobile.TelerikPushNotificationAdmin` API to allow an administrator to get, modify or delete notifications, send normal push notifications, and get the number of devices registered. The administrator would require an API key and the API Master key (both are available in the Telerik Platform backend settings). These values are passed to the constructor, with an optional URL. If a URL is not passed as an argument, the default URL of http://api.everlive.com/v1/ is used. It includes the following:
<table>
<thead>
<tr>
<th>Methods</th>
<th>Fully qualified name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetNotifications</td>
<td>public JsonObject GetNotifications(input poFilter as JsonObject) method public JsonObject GetNotifications()</td>
<td>Returns all notifications for the master key relevant to the provided filter. The data is returned as a JsonObject containing the notifications details, including an Id property. The filter argument is in JSON form.</td>
</tr>
<tr>
<td></td>
<td>Note: For more details about how to construct the filter, see the Telerik documentation (<a href="http://docs.telerik.com">http://docs.telerik.com</a>).</td>
<td></td>
</tr>
<tr>
<td>UpdateNotification</td>
<td>public void UpdateNotification(input pcNotificationId as character, input poUpdateValue as JsonObject).</td>
<td>Modifies the payload of a single notification. This method uses a notification Id (typically obtained from a GetNotifications call) and a JSON object containing the values to update in the notification.</td>
</tr>
<tr>
<td></td>
<td>Note: This JSON should be structured according to the format as specified in Telerik push-notifications structure</td>
<td></td>
</tr>
<tr>
<td>DeleteNotification</td>
<td>public void DeleteNotification(input pcNotificationId as character)</td>
<td>Deletes a notification from the server. The method uses a notification Id (typically obtained from a GetNotifications call).</td>
</tr>
<tr>
<td>GetDevices</td>
<td>public JsonObject GetDevices(input poFilter as JsonObject) method public JsonObject GetDevices()</td>
<td>Returns information about all devices for the master key relevant to the provided filter. The filter argument is in JSON form.</td>
</tr>
<tr>
<td></td>
<td>Note: For more details about how to construct the filter, see the Telerik documentation (<a href="http://docs.telerik.com">http://docs.telerik.com</a>).</td>
<td></td>
</tr>
<tr>
<td>GetDeviceCount</td>
<td>public integer GetDeviceCount()</td>
<td>Returns the count of all the devices that are currently registered for push notifications with Telerik Backend Services.</td>
</tr>
</tbody>
</table>
Here is an example of the `OpenEdge.Mobile.TelerikPushNotificationAdmin` API for getting notifications and viewing the number of devices registered:

```plaintext
BLOCK-LEVEL ON ERROR UNDO, THROW.
USING OpenEdge.Mobile.TelerikPushNotificationAdmin FROM PROPATH.

DEFINE VARIABLE oAdminSvc AS TelerikPushNotificationAdmin NO-UNDO.
DEFINE VARIABLE oResult AS JsonObject NO-UNDO.
DEFINE VARIABLE iLoop AS INTEGER NO-UNDO.

oAdminSvc = NEW TelerikPushNotificationAdmin(
    '<api key>', /* api */
    '<master key>' /* master */
).
oAdminSvc:Initialize().
oResult = oAdminSvc:GetNotifications(NEW JsonObject()).
oResult:writefile('notifications.json', TRUE).
iLoop = oAdminSvc:GetDeviceCount().
MESSAGE "Number of registered devices: " iLoop
VIEW-AS ALERT-BOX.
```
Introducing OpenEdge Business Rules

Progress Developer Studio for OpenEdge and Progress Corticon Studio are standalone Eclipse-based development environments that can be integrated into a single Eclipse instance to use the capabilities of integrated business rules in Progress OpenEdge. Progress Corticon is a Business Rules Management System. Its patented "no-coding" rules engine automates sophisticated decision processes. It provides the simple yet powerful Corticon Decision Services Methodology for modeling business rules.

Integrating OpenEdge and Corticon enables you to use ABL data structures (such as ProDataSets and temp-tables) as Corticon data structures (Vocabularies and entities). OpenEdge business rules provide tooling support for creating and updating Corticon Vocabularies, and runtime support for simple invocation of Corticon Decision Services.
You can use the integrated environment of Progress Developer Studio for OpenEdge and Corticon Studio to do the following:

- Use a Business Rules server to perform some or all of the business logic from an ABL client
- Re-use the existing business logic data structures (ProDataSets and/or temp-tables) with a Business Rules server without modification
- Generate output from a Business Rules server easily, using ABL data structures, instead of XML
- Generate a resource containing ProDataSet and temp-table schema information from an ABL source code
- Create ABL source code from a resource containing ProDataSet and temp-table schema information
- Import Corticon Vocabulary from a resource containing ProDataSet or temp-table schema information
- Export Corticon Vocabulary to a resource
- Test an OpenEdge application containing both ABL and Corticon business logic from within Progress Developer Studio for OpenEdge
- Create a single project in Progress Developer Studio for OpenEdge containing both ABL and Corticon artifacts

For details, see the following topics:

- Concepts
- Tasks
- Reference

## Concepts

### Terminology related to OpenEdge Business Rules

The following terms describe the various components that apply to an integrated development environment of OpenEdge and Corticon:

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProDataSet</td>
<td>A predefined view of data usually from multiple data sources (such as temp and database buffers) that are related to each other through data relationships. It is a potentially complex in-memory data structure that can be passed as a parameter with a single handle from one procedure to another, within a single OpenEdge session or between sessions.</td>
</tr>
<tr>
<td>Component</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Vocabulary</td>
<td>A structured dictionary containing all necessary business terms and relationships between them used by the Business Rules.                                                                                                  It is an eCore model that defines a relational data structure against which rules are applied. The relational data structure contains domains, entities, attributes and relationships (including cardinality), and maps to the ABL file’s ProDataSet and temp-tables. A Vocabulary is used to build rule models in a Rulesheet or test in a Ruletest.</td>
</tr>
<tr>
<td>Rulesheet</td>
<td>A set of rules (conditions and actions and plain language statements) written from a common business Vocabulary. By organizing these rules, it becomes a self-contained, independent unit of automated decision-making.</td>
</tr>
<tr>
<td>Ruleflow</td>
<td>A set of one or more Rulesheets organized for sequential execution.</td>
</tr>
<tr>
<td></td>
<td>A Ruleflow aggregates and organizes Rulesheets into a single unit of automated decision-making. It may be assembled from multiple Rulesheets, provided that the Rulesheets use the same Vocabulary file. In other words, a Ruleflow can only one associated Vocabulary. After a Ruleflow has been saved and deployed to the Corticon Server, it is called a Decision Service.</td>
</tr>
<tr>
<td>Ruletest</td>
<td>A mechanism within Corticon Studio for creating use cases or test scenarios of sample data and sending them to a Rulesheet or Ruleflow for processing. Ruletests consist of one or more Testsheets which test independent Rulesheets or Ruleflows, or can be linked together to test a succession of Rulesheets or Ruleflows to simulate a process sequence. Like Rulesheets, Ruletests also use a common Vocabulary model.</td>
</tr>
<tr>
<td>Decision Service</td>
<td>Automates a discrete decision-making task.</td>
</tr>
<tr>
<td></td>
<td>A Decision Service is implemented as a set of Business Rules and exposed as a web service or a Java service. By definition, the rules within a Decision Service are complete and unambiguous. For a given set of inputs, a Decision Service addresses every logical possibility uniquely, ensuring decision integrity. After they are deployed to the Corticon server, Ruleflows become Decision Services. Multiple versions of a single Decision Services can be deployed concurrently.</td>
</tr>
<tr>
<td>Business Rules server</td>
<td>A high-performance and scalable system resource that manages pools of Decision Services and executes their rules against incoming requests. It can be easily configured as a Web Services server that exposes Decision Services as true Web Services.</td>
</tr>
</tbody>
</table>

**See also**

[Introducing OpenEdge Business Rules](#) on page 313
Tasks

Integrating Corticon Studio and Progress Developer Studio for OpenEdge

You can use the capabilities of Business Rules in Progress OpenEdge by integrating Progress Developer Studio for OpenEdge and Progress Corticon Studio into a single environment.

To integrate Corticon Studio plugins to Progress Developer Studio for OpenEdge, use one of the following methods:

• While installing Corticon Studio, select the Eclipse location of Progress Developer Studio for OpenEdge as the external Eclipse location on the **External Eclipse configuration** page.

• After installing Progress Developer Studio for OpenEdge and Corticon Studio, open **Command Prompt**, go to the installation directory of Corticon Studio and run the following code:

  integrateCorticon.bat -install <PDSOE Eclipse location>

**Note:** For information on integrating Progress Developer Studio for OpenEdge plugins to Corticon Studio, see the **Progress Corticon Documentation**.

See also

Introducing OpenEdge Business Rules on page 313

Configuring Corticon Server on the OE Web Server

OpenEdge is shipped with the OE Web Server. The Tomcat server is available in your OpenEdge installation directory `%DLC%/servers/tomcat`. It provides a Java container and a web server.

You must configure the Corticon server on your Tomcat server to make it available for deploying projects.

By default, Corticon Server is a part of the Corticon Studio installation but has the following license restrictions:

• The request must come from the same machine on which the server is running. Requests from any other machine or host are disallowed.

• Only a single reactor is available per Decision Service. This means that only one Decision Service can be executed at a time. This allows you to deploy multiple Decision Services simultaneously. Since ABL is single-threaded and since this license is intended for a single-user development scenario, it is ideal for the intended use case. If you want to test the system more thoroughly, you must purchase a proper licensed server.

**Note:** These conditions only apply to the execution of a Decision Service. They do not affect the service availability check and other administrator functions.

To configure the Corticon server:

1. Based on the Corticon Server (purchased or default) that you are using, do one of the following:
• If you are using the default Corticon server that comes with the Corticon Studio installation, copy the
  Axis.war file located in the Corticon Studio installation directory to the OpenEdge
  %DLC%\servers\tomcat\webapps directory.
• If you have purchased the Corticon server license, download the Corticon Server for Java Archive from
  the Progress Download Center available at http://www.progress.com/esd, and copy the Axis.war file
  located in it to the OpenEdge %DLC%\servers\tomcat\webapps directory.

2. Create a batch file in the OpenEdge work directory (%DLC%\WRK) and specify its name.
3. Open the batch file and set the value of the JRE_HOME variable as: SET JRE_HOME=%DLC%\jdk\jre
4. Start the OpenEdge Web Server.
   a) Open the Servers view in an OpenEdge perspective of Progress Developer Studio for OpenEdge by
      selecting Window > Show View > Other > Server > Servers > OK.
   b) Select the REST Manager instance and then, select Start in the context menu or on the Servers toolbar.

   Note: You can also start the OpenEdge Web Server from the Proenv command prompt using protc
      start.

5. In a web browser, enter the URL for the OpenEdge Web Server with default ports:
   http://localhost:8980/axis, and go to the Corticon Server console to verify if the Corticon Server
   is successfully installed.

   The Corticon Server Console login page appears after successful installation. The Corticon server has
   the following six defined server console login user names:

<table>
<thead>
<tr>
<th>User name</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin</td>
<td>admin</td>
</tr>
<tr>
<td>administrator</td>
<td>changeme</td>
</tr>
<tr>
<td>modeler2</td>
<td>modeler2</td>
</tr>
<tr>
<td>modeler1</td>
<td>modeler1</td>
</tr>
<tr>
<td>Tester</td>
<td>tester</td>
</tr>
</tbody>
</table>

   Note: See the Progress Corticon Documentation for more information about using the Corticon server and
   for other installation options.

Security

Securing the OE Web Server is part of the process of configuring the Corticon Server, especially if it runs
within the OE Web Server. Since Corticon Server runs as an independent WebApp, you can secure access
  to the Rules Service using the security models available in the OE Web Server. For more information on
  securing the OE Web Server, see the OpenEdge Development: Mobile Applications book.
Generating Business Rules Vocabulary Definition

A Business Rules Vocabulary Definition (.brvd) file is used to exchange ProDataSet and/or temp-table schema information between OpenEdge and Corticon at design time. It is a proprietary format of an XML-based file.

The schema generated and exported from OpenEdge can be imported and used as the basis for Vocabulary entities and attributes in Corticon Studio. For more information on importing business rules vocabulary definition, see the *Progress Corticon Documentation*.

To generate a Business Rules Vocabulary Definition file:

1. Create an OpenEdge project or select an existing one.
2. From the project or editor's context (right-click) menu, select *Progress OpenEdge > Generate Business Rules Vocabulary Definition*.

   **Note:** You can also select the *Generate Business Rules Vocabulary Definition* menu option from the *Outline* view. See *Using Outline view to generate business rules vocabulary definition* for more information.

   The *Generate Business Rules Vocabulary Definition* wizard appears.

3. In the left tab, select the ABL source file whose ProDataSets and/or temp-tables you want to add to the Business Rules Vocabulary Definition file.

4. By default, the wizard displays the *Workspace Resources* tab that lists files in the workspace. To view files in the file system, select the drop-down arrow and then select *File System*.

   **Note:** You can use the *Browse ABL Files* option available on the toolbar of the *File System Resources* tab to add files from the local file system to the current list. This option is enabled only when you select *File System*. To use database temp-tables containing the LIKE keyword you must select a workspace resource.

5. When you select a file in the left tab, the *ProDataSets and Temp-Tables* tab displays all the ProDataSets and temp-tables available in that file. Select the ProDataSet and/or temp-table that you want to include in the Business Rules Vocabulary Definition file.

   **Note:** If the selected file has any compilation errors, then no ProDataSets and temp-tables are displayed for that file.

   The *Selected data structures* list displays the ProDataSets and/or temp-tables that you select in the *ProDataSets and Temp-Tables* tab. To remove an item from this list, select the item and then select on the *Selected data structures* toolbar.

6. In the *Definition file* field, click *Browse* to specify a location in your file system and a name for the Business Rules Vocabulary Definition file.
Note: If you have specified a default rules Vocabulary folder on the Business Rules preferences page, the location of that folder is displayed when you click Browse. You can change this location if necessary. You can also specify a workspace location for the Business Rules Vocabulary Definition file. In which case, you must refresh the OpenEdge project containing the file to be able to view it in the Project Explorer view.

7. Click Finish.

A Business Rules Vocabulary Definition (.brvd) file is created in the specified location with the selected ProDataSets and/or temp-tables. If the same file already exists, a dialog prompts you to replace it.

See also
Exporting Business Rules Vocabulary Definition on page 319
Generate Business Rules Vocabulary Definition wizard on page 323

Using Outline view to generate business rules vocabulary definition

You can use the Outline view to open the Business Rules Vocabulary Definition wizard to generate a Business Rules Vocabulary Definition file.

1. Open the procedure file in the Project Explorer view.
2. Select Window > Show View > Outline.

   The Outline view opens and displays all the defined ProDatasets or a TempTables of the procedure file that you have selected.
3. Select a ProDataset or a TempTable, right-click and select Generate Business Rules Vocabulary Definition.

   The Generate Business Rules Vocabulary Definition wizard opens.

   For more information on generating business rules Vocabulary definition, see Generating Business Rules Vocabulary Definition on page 318.

Exporting Business Rules Vocabulary Definition

A Business Rules Vocabulary Definition (.brvd) file is used to exchange ProDataSet and/or temp-table schema information between OpenEdge and Corticon at design time. It is a proprietary format of an XML-based file.

The schema generated and exported from OpenEdge can be imported and used as the basis for Vocabulary entities and attributes in Corticon Studio. For more information on importing business rules vocabulary definition, see the Progress Corticon Documentation.

To export a Business Rules Vocabulary Definition file:

1. Select File > Export > Progress OpenEdge > Business Rules Vocabulary Definition.

   Note: You can also select this option from the context (right-click) menu in the Project Explorer view.

   The Export Business Rules Vocabulary Definition wizard appears.
2. In the left tab, select the ABL source file whose ProDataSets and/or temp-tables you want to add to the Business Rules Vocabulary Definition file.
3. By default, the wizard displays the Workspace Resources tab that lists files in the workspace. To view files in the file system, select the drop-down arrow and then select File System.
Note: You can use the Browse ABL Files option available on the toolbar of the File System Resources tab to add files from the local file system to the current list. This option is enabled only when you select File System. To use database temp-tables containing the LIKE keyword you must select a workspace resource.

4. When you select a file in the left tab, the ProDataSets and Temp-Tables tab displays all the ProDataSets and temp-tables available in that file. Select the ProDataSet and/or temp-table that you want to include in the Business Rules Vocabulary Definition file.

Note: If the selected file has any compilation errors, then no ProDataSets and temp-tables are displayed for that file.

The Selected data structures list displays the ProDataSets and/or temp-tables that you select in the ProDataSets and Temp-Tables tab. To remove an item from this list, select the item and then select on the Selected data structures toolbar.

5. In the Definition file field, click Browse to specify a location in your file system and a name for the Business Rules Vocabulary Definition file.

Note: If you have specified a default rules Vocabulary folder on the Business Rules preferences page, the location of that folder is displayed when you click Browse. You can change this location if necessary. You can also specify a workspace location for the Business Rules Vocabulary Definition file. In which case, you must refresh the OpenEdge project containing the file to be able to view it in the Project Explorer view.

6. Click Finish.

The selected ProDataSets and/or temp-tables are exported to the specified Business Rules Vocabulary Definition file.

See also
Generating Business Rules Vocabulary Definition on page 318
Export Business Rules Vocabulary Definition wizard on page 322

Setting Business Rules preferences

The Business Rules preferences page allows you to specify the default folder where the Business Rules Vocabulary Definition (.brvd) file is located.

To set preferences for Business Rules:

1. Select Window > Preferences > Progress OpenEdge > Business Rules.

2. At the Rules vocabulary folder field, select Browse to specify the location where you want to create the Business Rules Vocabulary Definition file.

The location that you specify here becomes the default location of the Business Rules Vocabulary Definition file and is displayed when you select the Browse button at the Definition file field of the Generate Business Rules Vocabulary Definition and Export Business Rules Vocabulary Definition wizards. You can change this location at the time of generating or exporting the Business Rules Vocabulary Definition file.
Using Business Rules API to invoke Decision Services

OpenEdge allows you to use Business Rules API to call the Decision Service that is deployed onto the Corticon Server. A SOAP message is sent to the Corticon Server and Corticon updates the values of the entity and returns them to OpenEdge. The returned Vocabulary model structure is the same as the one passed.

**Note:** To invoke a Decision service from ABL business logic, you must add the required procedure library and include files to the project's PROPATH. See Adding Business Rules libraries to PROPATH for more information.

To invoke a Decision Service, use the following API classes:

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenEdge.BusinessRules.RulesServerConnectionParameters</td>
<td>Returns parameters for connecting to a Business Rules server. Initially, access to Decision Service and Admin Service are through separate connections (although they can be specified by one parameter).</td>
</tr>
<tr>
<td>OpenEdge.BusinessRules.RulesServerConnection</td>
<td>Describes the server connection for an OpenEdge Rules Server. This is a specialized class whose main purpose is to provide an abstraction from the fact that the connection to the Rules server is a Web Services connection.</td>
</tr>
<tr>
<td>Progress.Json.ObjectModel.JsonObject</td>
<td>Denotes a dynamic number of properties, each addressable by a Unicode string called a name.</td>
</tr>
<tr>
<td>Progress.Lang.AppError</td>
<td>Is the ultimate super class of all application errors. An application error is simply any collection of data you need to provide meaningful information about a condition. Representing a user-defined error as a error object allows your application to throw and catch or return the error in the ABL structured error handling model.</td>
</tr>
</tbody>
</table>

Adding Business Rules libraries to PROPATH

You can configure an OpenEdge project with Corticon settings to run Corticon-related functionality from within Progress Developer Studio for OpenEdge. To do so, you must add the Business Rules libraries (OpenEdge.BusinessRules.pl file) to the list of procedure libraries appended to PROPATH in Progress Developer Studio for OpenEdge.

**Note:** You must also add these entries to the PROPATH of any production environments.
To add the Business Rules libraries to PROPATH:

1. Select a project in the Project Explorer view and select Properties from the context (right-click) menu or select Project > Properties from the main menu bar.
   The Properties page for the selected project appears.

2. Select Progress OpenEdge > PROPATH.

   **Note:** For a shared AVM project, select Window > Preferences > Progress OpenEdge > Shared AVM > PROPATH.

3. Select Add External Library and select the OpenEdge.BusinessRules.pl file from the file system.

   **Note:** If the project uses GUI as the runtime, the OpenEdge.BusinessRules.pl file is located in @{DLC}\gui\rules and if the project uses TTY as the runtime, the file is located in @{DLC}\tty\rules, where DLC is the default installation directory.

   The selected procedure library appears in the PROPATH section (the left pane on the PROPATH properties page.

4. To use the output of the GetMessages() API (which, returns the messages for the most recent decision service invocation), add the @{DLC}\gui\rules or @{DLC}\tty\rules folder to the PROPATH along with the OpenEdge.BusinessRules.pl file by selecting Add External Directory on the PROPATH properties page.

5. Click OK.

   **Note:** You can also add PROPATH entries while creating a project using the Define PROPATH page.

**See also**

Using Business Rules API to invoke Decision Services on page 321

**Reference**

**Export Business Rules Vocabulary Definition wizard**

The Export Business Rules Vocabulary Definition wizard allows you to export ProDataSets and temp-tables from the workspace and/or file system to an intermediate Business Rules Vocabulary definition (.brvd) file. This intermediate .brvd file can be imported to Corticon Studio to generate and package Vocabulary and rules and then deployed onto the Corticon server.

The Export Business Rules Vocabulary Definition wizard includes the following controls:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workspace Resources</td>
<td>(Default) Displays a list of ABL class and procedure files available in workspace.</td>
</tr>
</tbody>
</table>
### Generate Business Rules Vocabulary Definition wizard

The **Generate Business Rules Vocabulary Definition** wizard allows you to generate a Business Rules Vocabulary Definition (.brvd) file for the selected ProDataSets or temp-tables. This intermediate .brvd file can be imported to Corticon Studio to generate and package Vocabulary and rules and then be deployed onto the Corticon server.

The **Generate Business Rules Vocabulary Definition** wizard includes the following controls:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workspace Resources</td>
<td>(Default) Displays a list of ABL class and procedure files available in workspace.</td>
</tr>
<tr>
<td>File System Resources</td>
<td>Displays a list of files that you select from the file system. This tab appears when you select <strong>File System</strong> using the drop-down arrow on the toolbar.</td>
</tr>
<tr>
<td>ProDataSets and Temp-Tables</td>
<td>Displays the available ProDataSets and temp-tables for each selected file.</td>
</tr>
<tr>
<td>Selected data structures</td>
<td>Displays the ProDataSets and temp-tables that you select in the <strong>ProDataSets and Temp-Tables</strong> tab.</td>
</tr>
<tr>
<td>Definition file</td>
<td>Specifies the location where the Business Rules Vocabulary Definition file is saved.</td>
</tr>
</tbody>
</table>

**See also**

- Generating Business Rules Vocabulary Definition on page 318

---

### OpenEdge Business Rules API reference

The following sections describe the various APIs supported by the OpenEdge-Corticon framework for invoking Decision Services:

---

**See also**

- Exporting Business Rules Vocabulary Definition on page 319

---

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workspace Resources</td>
<td>(Default) Displays a list of ABL class and procedure files available in workspace.</td>
</tr>
<tr>
<td>File System Resources</td>
<td>Displays a list of files that you select from the file system. This tab appears when you select <strong>File System</strong> using the drop-down arrow on the toolbar.</td>
</tr>
<tr>
<td>ProDataSets and Temp-Tables</td>
<td>Displays the available ProDataSets and temp-tables for each selected file.</td>
</tr>
<tr>
<td>Selected data structures</td>
<td>Displays the ProDataSets and temp-tables that you select in the <strong>ProDataSets and Temp-Tables</strong> tab.</td>
</tr>
<tr>
<td>Definition file</td>
<td>Specifies the location where the Business Rules Vocabulary Definition file is saved.</td>
</tr>
</tbody>
</table>
Class and Interface reference

This section contains reference entries that describe the built-in classes and interfaces supported for OpenEdge Business Rules. These classes and interfaces contain types and other artifacts directly pertaining to OpenEdge and Corticon integration. They also contain types and other artifacts for generic handling of server connections and their parameters.

OpenEdge.BusinessRules.DecisionService class

The OpenEdge.BusinessRules.DecisionService class provides the functionality to create a Decision Service instance.

Constructors

The following constructor invokes the currently available version of a Decision Service, as determined by the Corticon Server, specified by the OpenEdge.BusinessRules.RulesServerConnection class:

```public
PUBLIC (pcDecisionServiceName AS CHARACTER, poServer AS OpenEdge.BusinessRules.RulesServerConnection)
```

The following constructor invokes the specified version of a Decision Service running on the Corticon Server as specified by the OpenEdge.BusinessRules.RulesServerConnection class:

```public
```

The following constructor invokes the version of a Decision Service running on the Corticon Server that is effective at the date or time provided, as specified by the OpenEdge.BusinessRules.RulesServerConnection class:

```public
```

Super Class

Progress.Lang.Object class

Public Properties

<table>
<thead>
<tr>
<th>Name property</th>
<th>Version property</th>
<th>EffectiveOn property</th>
</tr>
</thead>
</table>

Protected Property

ServerConnection property

Public Methods

| InvokeService() method | GetMessages() method | IsServiceAvailable() method |
OpenEdge.BusinessRules.RulesServerConnection class

The `OpenEdge.BusinessRules.RulesServerConnection` class describes the server connection for an OpenEdge Business Rules server. It uses Web Services connections to access the Business Rules server.

Super Class

`OpenEdge.Core.ServerConnection.WebServiceConnection` class

OpenEdge.BusinessRules.RulesServerConnectionParameters class

The `OpenEdge.BusinessRules.RulesServerConnectionParameters` class returns parameters for connecting to a Business Rules server. Although the Decision Service and Administration Service can be specified by one parameter, access to them is through separate connections. The initial implementation uses Web Services but the `OpenEdge.BusinessRules.DecisionService` class abstracts that from a caller. The implementation affects the parameters values passed in.

Constructors

The following constructor creates a JSON object containing valid connection parameters:

```java
PUBLIC (INPUT poOptions AS JsonObject)
```

The following constructor creates a character representation of a JSON object containing valid connection parameters:

```java
PUBLIC (INPUT pcOptions AS CHARACTER)
```

Interface

`OpenEdge.Core.ServerConnection.IConnectionParameters` interface

Public Properties

<table>
<thead>
<tr>
<th>DecisionServiceConnectionParameters property</th>
<th>AdminServiceConnectionParameters property</th>
<th>FormatMask property</th>
</tr>
</thead>
</table>

Public Method

`GetConnectionString()` method

OpenEdge.Core.ServerConnection.FormatMaskEnum class

The `OpenEdge.Core.ServerConnection.FormatMaskEnum` class contains various simple format masks for connection parameters. These format masks can also be represented by individual format provider classes.

Super Class

`OpenEdge.Lang.EnumMember` class
Public Properties

<table>
<thead>
<tr>
<th>Default property</th>
<th>None property</th>
<th>DashSpace property</th>
<th>ABLConnect property</th>
</tr>
</thead>
<tbody>
<tr>
<td>DoubleDashSpace property</td>
<td>DoubleDashEquals property</td>
<td>NameEquals property</td>
<td>Custom property</td>
</tr>
</tbody>
</table>

**OpenEdge.Core.ServerConnection.IconnectionParameters interface**

The `OpenEdge.Core.ServerConnection.IconnectionParameters` interface returns connection parameters for the Decision Service calls.

**Public Property**

<table>
<thead>
<tr>
<th>FormatMask property</th>
</tr>
</thead>
</table>

**Public Method**

<table>
<thead>
<tr>
<th>GetConnectionString() method</th>
</tr>
</thead>
</table>

**See also**

`OpenEdge.BusinessRules.RulesServerConnectionParameters class` on page 325

**OpenEdge.Core.ServerConnection.IServerConnection interface**

The `OpenEdge.Core.ServerConnection.IServerConnection` interface describes the connection to a server (such as a database, an AppServer, or a Web Service).

**Public Properties**

<table>
<thead>
<tr>
<th>Server property</th>
<th>ConnectionParameters property</th>
<th>Connected property</th>
</tr>
</thead>
</table>

**Public Methods**

<table>
<thead>
<tr>
<th>Connect() method</th>
<th>Disconnect() method</th>
<th>CreateServer() method</th>
<th>DestroyServer() method</th>
</tr>
</thead>
</table>

**OpenEdge.Core.ServerConnection.WebServiceServerConnection class**

The `OpenEdge.Core.ServerConnection.WebServiceServerConnection` class describes a Web Service connection.

**Constructors**

```plaintext
PUBLIC (INPUT poConnectionParameters AS OpenEdge.Core.ServerConnection.WebServiceServerConnectionParameters)
```

**Interface**

`OpenEdge.Core.ServerConnection.IServerConnection interface`
OpenEdge.Lang.EnumMember class

The OpenEdge.Lang.EnumMember class is an enumeration definition class. It is an abstract class.

Constructors

The following constructor allows the specification of an integer value and a character name for the enumeration:

```plaintext
PROTECTED (piValue as INTEGER INPUT pcName AS CHARACTER)
```

The following constructor allows only the specification of an integer value for the enumeration:

```plaintext
PROTECTED (piValue as INTEGER)
```

The following constructor allows only the specification of a character name for the enumeration:

```plaintext
PROTECTED (INPUT pcName AS CHARACTER)
```

Public Properties

<table>
<thead>
<tr>
<th>Name property</th>
<th>Value property</th>
</tr>
</thead>
</table>

Public Method

ToString() method

OpenEdge.Lang.WidgetHandle class

The OpenEdge.Lang.WidgetHandle class is a primitive class for widget-handle variables.

Constructors

The following default constructor results in the WidgetHandle object having an unknown value:

```plaintext
PUBLIC ()
```

The following constructor initializes the WidgetHandle's value:

```plaintext
PUBLIC (phValue AS HANDLE)
```

Super Class

Progress.Lang.Object class
### Public Property

| Value property |

### Public Method

| ToString() method |

## Class Properties and Methods reference

This section contains reference entries that describe each built-in class property and method that ABL supports for working with OpenEdge Business Rules and structured error handling. Class properties and methods are mechanisms that allow you to monitor and control the behavior of class-based objects.

### ABLConnect property

The ABLConnect property defaults to `OpenEdge.Core.ServerConnection.FormatMaskEnum:DashSpace`.  
**Data type:** `OpenEdge.Core.ServerConnection.FormatMaskEnum`  
**Access:** PUBLIC STATIC  
**Applies to:** `OpenEdge.Core.ServerConnection.FormatMaskEnum` class

### AdminServiceConnectionParameters property

The AdminServiceConnectionParameters property contains connection parameters for the Admin Service calls such as `IsServiceAvailable()`.  
**Data type:** `OpenEdge.Core.ServerConnection.IConnectionParameters`  
**Access:** PUBLIC  
**Applies to:** `OpenEdge.BusinessRules.RulesServerConnectionParameters` class

### Connected property

The Connected property returns `True` if the server is valid and connected. Otherwise, returns `False`.  
**Data type:** LOGICAL  
**Access:** PUBLIC  
**Applies to:** `OpenEdge.Core.ServerConnection.IServerConnection` interface

### ConnectionParameters property

The ConnectionParameters property specifies the parameters for the connection.  
**Data type:** `OpenEdge.Core.ServerConnection.IConnectionParameters`  
**Access:** PUBLIC  
**Applies to:** `OpenEdge.Core.ServerConnection.IServerConnection` interface

### Custom property

The Custom property is a custom mask that is too complex for simple substitute-based parsing.  
**Data type:** `OpenEdge.Core.ServerConnection.FormatMaskEnum`
DashSpace property
The DashSpace property resolves to '-&1 &2'.
Data type: OpenEdge.Core.ServerConnection.FormatMaskEnum
Access: PUBLIC STATIC
Applies to: OpenEdge.Core.ServerConnection.FormatMaskEnum class

DecisionServiceConnectionParameters property
The DecisionServiceConnectionParameters property contains connection parameters for the Decision Service calls.
Data type: OpenEdge.Core.ServerConnection.IConnectionParameters
Access: PUBLIC
Applies to: OpenEdge.BusinessRules.RulesServerConnectionParameters class

Default property
The Default property defaults to OpenEdge.Core.ServerConnection.FormatMaskEnum:DashSpace.
Data type: OpenEdge.Core.ServerConnection.FormatMaskEnum
Access: PUBLIC
Applies to: OpenEdge.Core.ServerConnection.FormatMaskEnum class

DoubleDashEquals property
The DoubleDashEquals property resolves to '--&1=&2'.
Data type: OpenEdge.Core.ServerConnection.FormatMaskEnum
Access: PUBLIC STATIC
Applies to: OpenEdge.Core.ServerConnection.FormatMaskEnum class

DoubleDashSpace property
The DoubleDashSpace property resolves to '--&1 &2'.
Data type: OpenEdge.Core.ServerConnection.FormatMaskEnum
Access: PUBLIC STATIC
Applies to: OpenEdge.Core.ServerConnection.FormatMaskEnum class

EffectiveOn property
The EffectiveOn property is an optional property that specifies the effective date for selecting the Decision Service to be invoked. It defaults to an unknown value, which is the latest or the most recent version.

Note: Either the Version or EffectiveOn property can be used at a time, which you can decide when you call the constructor.
Data type: DATETIME-TZ
Access: PUBLIC
Applies to: OpenEdge.BusinessRules.DecisionService class

FormatMask property
The FormatMask property must always be set to FormatMaskEnum:Custom for WebService and URL connection parameters.

Data type: OpenEdge.Core.ServerConnection.FormatMaskEnum
Access: PUBLIC
Applies to: OpenEdge.BusinessRules.RulesServerConnectionParameters class; OpenEdge.Core.ServerConnection.IConnectionParameters interface

Name property
The Name property specifies the name of the Decision Service to be invoked. This value is mandatory and must be entered as a constructor argument.

Data type: CHARACTER
Access: PUBLIC
Applies to: OpenEdge.BusinessRules.DecisionService class; OpenEdge.Lang.EnumMember class

NameEquals property
The NameEquals property resolves to '&1=&2'.

Data type: OpenEdge.Core.ServerConnection.FormatMaskEnum
Access: PUBLIC STATIC
Applies to: OpenEdge.Core.ServerConnection.FormatMaskEnum class

None property
The None property does not require a default format mask to be applied.

Data type: OpenEdge.Core.ServerConnection.FormatMaskEnum
Access: PUBLIC STATIC
Applies to: OpenEdge.Core.ServerConnection.FormatMaskEnum class

Operation property
The Operation property is extracted from the WebServiceConnectionParameters object passed in to the class.

Data type: CHARACTER
Access: PUBLIC
Applies to: OpenEdge.Core.ServerConnection.WebServiceServerConnection class

PortName property
The PortName property is extracted from the WebServiceConnectionParameters object passed in to the class.
**Server property**

The Server property refers to the actual server. The object acts as a wrapper for handle-based servers. For `OpenEdge.Core.ServerConnection.WebServiceServerConnection` class, the Server property is of type `OpenEdge.Lang.WidgetHandle`.

**Data type:** CHARACTER  
**Access:** PUBLIC  
**Applies to:** `OpenEdge.Core.ServerConnection.WebServiceServerConnection` class

---

**ServerConnection property**

The value for the Version property is mandatory and must be entered as constructor argument.

**Data type:** `OpenEdge.Rules.RulesServerConnection`  
**Access:** PROTECTED  
**Applies to:** `OpenEdge.BusinessRules.DecisionService` class

---

**ServiceName property**

The ServiceName is extracted from the `WebServiceConnectionParameters` object passed in to the class.

**Data type:** CHARACTER  
**Access:** PUBLIC  
**Applies to:** `OpenEdge.Core.ServerConnection.WebServiceServerConnection` class

---

**Value property**

The Value property is set through a constructor.

**Data type:** INTEGER  
**Access:** PUBLIC STATIC  
**Applies to:** `OpenEdge.Lang.EnumMember` class; `OpenEdge.Lang.WidgetHandle` class

---

**Version property**

The Version property is an optional property that specifies the version of the Decision Service to be invoked. It defaults to an unknown value, which is the latest or the most recent version.

**Note:** Either the Version or EffectiveOn property can be used at a time, which you can decide when you call the constructor.

**Data type:** DECIMAL  
**Access:** PUBLIC  
**Applies to:** `OpenEdge.BusinessRules.DecisionService` class
CreateServer() method

The CreateServer() method creates a server object. This method is separated from the Connect() and Disconnect() methods so that a server can be connected and disconnected multiple times.

Return type: VOID
Access: PUBLIC
Applies to: OpenEdge.Core.ServerConnection.IServerConnection interface
Syntax

CreateServer()

Connect() method

The Connect() method connects to the specified server based on the ConnectionParameters.

Return type: VOID
Access: PUBLIC
Applies to: OpenEdge.Core.ServerConnection.IServerConnection interface
Syntax

Connect()

DestroyServer() method

The DestroyServer() method deletes a server object. This method is separated from the Connect() and Disconnect() methods so that a server can be connected and disconnected multiple times.

Return type: VOID
Access: PUBLIC
Applies to: OpenEdge.Core.ServerConnection.IServerConnection interface
Syntax

DestroyServer()

Disconnect() method

The Disconnect() method disconnects from the server if connected.

Return type: VOID
Access: PUBLIC
Applies to: OpenEdge.Core.ServerConnection.IServerConnection interface
### Syntax

```
Disconnect()
```

### GetConnectionString() method

**Return type:** CHARACTER  

**Access:** PUBLIC  

**Applies to:** OpenEdge.BusinessRules.DecisionService class;  
OpenEdge.Core.ServerConnection.IConnectionParameters interface

The following version of the method overrides and returns the connection string from the DecisionServiceConnectionParameters property:

```
GetConnectionString()
```

The following version of the method overrides and returns the connection string from the DecisionServiceConnectionParameters property. If the ConnectionParameters object has a custom format mask, an AppError is returned.

```
GetConnectionString(INPUT pcFormatMask AS CHARACTER)
```

The following version of the method overrides and returns the connection string from the DecisionServiceConnectionParameters property. If the ConnectionParameters object has a custom format mask, an AppError is returned.

```
GetConnectionString(INPUT poFormatMask AS FormatMaskEnum)
```

### GetMessages() method

The **GetMessages()** method returns the messages for the most recent Decision Service invocation. It is an idempotent operation, so a repeated call to this method always returns the same data. The message data is only cleared when a new InvokeService call is made. The RulesMessage table refers to the response data structure used in the InvokeService() call. These references may differ from the input application data structure. The GetMessages() method always returns at least one row containing the version of the Decision Service that was used (for cases where an EffectiveOn value or no version was specified). The message has a severity of **Info** and displays a message text specifying the version of Decision Service that was invoked. The remaining fields contain unknown values.

**Return type:** VOID  

**Access:** PUBLIC  

**Applies to:** OpenEdge.BusinessRules.DecisionService class
Syntax

```plaintext
GetMessages(OUTPUT TABLE RulesMessage)
```

**InvokeService() method**

**Return type:** VOID  
**Access:** PUBLIC  
**Applies to:** OpenEdge.BusinessRules.DecisionService class

The following version of the method accepts the table handle to be passed to the Decision Service. You must call this method BY-REFERENCE for a shallow copy. The contents of this temp-table are removed and replaced by the results of the Decision Service call. The temp-table passed into this method cannot be part of a ProDataSet, otherwise an AppError is raised.

```plaintext
InvokeService(INPUT-OUPUT TABLE-HANDLE phAppData)
```

The following overloaded version of the method accepts the table handle to be passed to the Decision Service. You must call this method BY-REFERENCE for a shallow copy. The results of the Decision Service call are returned in the output parameter without updating the input data. The temp-table passed into this method cannot be part of a ProDataSet, otherwise an AppError occurs. The output table is a copy of the input table. The intent of this API is to allow callers to use the input data as a before-image for comparison purposes. The RulesMessage table returned by the GetMessages() method contains a field named DataKeyValue.

```plaintext
InvokeService(INPUT TABLE-HANDLE phAppData, OUTPUT TABLE-HANDLE phResponseData)
```

The following version of the method accepts the ProDataSet handle to be passed to the Decision Service. You must call this method BY-REFERENCE for a shallow copy. The contents of this ProDataSet are removed and replaced by the results of the Decision Service call.

```plaintext
InvokeService(INPUT-OUPUT DATASET-HANDLE phAppData)
```

The following version of the method specifies the ProDataSet handle to be passed to the Decision Service. You must call this method BY-REFERENCE for a shallow copy. The results of the Decision Service call are returned in the output parameter without updating the input data. The output data set is a copy of the input data set. The intent of this API is to allow callers to use the input data as a before-image for comparison purposes. The RulesMessage table returned by the GetMessages() method contains a field named DataKeyValue that refers to the tables in the response ProDataSet.

```plaintext
InvokeService(INPUT DATASET-HANDLE phAppData, OUTPUT DATASET-HANDLE phResponseData)
```

**IsServiceAvailable() method**

The IsServiceAvailable() method determines whether the Decision Service is available or not.
**Return type:** LOGICAL  
**Access:** PUBLIC  
**Applies to:** OpenEdge.BusinessRules.DecisionService class

**Syntax**

```plaintext
IsServiceAvailable()
```

**ToString() method**

If applied to the OpenEdge.Lang.EnumMember class, the ToString() method returns the Name property. If the Name property is unknown, it returns the Value property.

**Return type:** CHARACTER  
**Access:** PUBLIC  
**Applies to:** OpenEdge.Lang.EnumMember class; OpenEdge.Lang.WidgetHandle class

**Syntax**

```plaintext
ToString()
```
Introducing WebSpeed support in Progress Developer Studio for OpenEdge

This section of the Progress Developer Studio for OpenEdge Guide describes the tools and features in the Progress Developer Studio that support the WebSpeed application development environment.

WebSpeed® is an ABL environment for developing and deploying Web applications. Typically, web applications are applications that accessed through a Web browser. WebSpeed applications usually involve some interaction with a data source. In WebSpeed, you can implement queries, updates, and the addition or deletion of records.

With WebSpeed, you can develop and deploy:

- Intranet applications that allow internal users to access and modify data.
- Internet applications that allow external, consumer access (for example, shopping cart applications).
- Extranet, business-to-business applications.

Progress Developer Studio for OpenEdge includes a WebSpeed project type that supports the Embedded SpeedScript and CGI Wrapper programming models. (HTML Mapping is supported indirectly in the AppBuilder running embedded in an Eclipse window.) In addition, Progress Developer Studio implements Eclipse publishing functionality, which allows you to copy and compile your code on a remote or local server with a single mouse click.

**Note:** WebSpeed project type can be deployed only to the traditional servers, to deploy to PAS for OpenEdge, create ABL Web App projects.

You will find that Progress Developer Studio for OpenEdge, as an integrated development environment, also has many advantages over the loose collection of OpenEdge tools known as WebSpeed Workshop, which includes the browser-based utility, WebTools. Note that some features in WebTools are not currently supported in Progress Developer Studio, namely:
• Scripting lab
• Viewing CGI and other session variables
• Individual file level compilation
• Directory browsing
• Remote file editing

However, you can launch WebTools directly from Progress Developer Studio if you need to access any of these features. (Right-click on a WebSpeed Server in the Server view and select Launch WebSpeed Workshop from the context menu.)

For more information about WebSpeed itself and examples of WebSpeed applications, see the following manuals in the OpenEdge Product documentation:

• OpenEdge Getting Started: WebSpeed Essentials
• OpenEdge Getting Started: Application and Integration Services
• OpenEdge Application Server: Developing WebSpeed Applications
• OpenEdge Application Server: Administration

For details, see the following topics:

• Concepts
• Tasks
• Reference

Concepts

This section contains some general information about WebSpeed components and architecture.

WebSpeed architecture

The architecture of WebSpeed is designed to:

• Receive requests, in the form of a URL, from a client on the Web.
• Process the request, which often includes interaction with a data source.
• Create an HTML page as a reply to the request.
• Pass the HTML page back to the client.

The following diagram shows the basic structure of WebSpeed and how its components interact to handle a request from a client.
From the diagram, you can see that the WebSpeed Messenger is a key component of the architecture. As the diagram implies, it is the only WebSpeed component that must be installed on the same machine as the Web server. All other WebSpeed components can be installed on the same, or on a different machine from the Web server. The Messenger is usually installed in an appropriate scripts directory (for example, /cgi-bin) on the Web server.

During an OpenEdge installation, you specify a WebSpeed Messenger that is compatible with your Web server. (Note that the Web server is not a component of the WebSpeed product.) Web server types supported include ISAPI-compatible (Microsoft IIS), NSAPI-compatible (Sun Web server), or CGI-compatible (almost any Web server including Apache.) There is also a Messenger that works with Microsoft’s Active Server Pages, the WSASP Messenger.

As the diagram implies, a client initiates a WebSpeed request that is received by a Web server and passed to the Messenger. The Messenger contacts the NameServer to get the name of a WebSpeed Broker that has the resources to handle the request. The request is then passed to an appropriate Broker. The Broker finds a WebSpeed Agent that can run the code (Web objects) that services the request. The output is formatted as HTML, returned to the Messenger, and sent to the HTTP Client via the Web server.

Notice that the AdminServer is not directly involved with servicing a WebSpeed request. The AdminServer is an administrative framework that starts and stops components based on configuration information contained in the ubroker.properties file.

This topic is a simplified version of the WebSpeed architecture and how it functions. For more information, see the following manuals in the OpenEdge Product documentation:

- OpenEdge Getting Started: WebSpeed Essentials
- OpenEdge Application Server: Administration

See also
WebSpeed and Eclipse server terminology on page 340
Web objects on page 340
Standard HTML files on page 341
Embedded SpeedScript files on page 341
CGI Wrapper files on page 342
WebSpeed and Eclipse server terminology

The following list explains some of the server terminology used in the documentation for WebSpeed application development in Progress Developer Studio for OpenEdge:

- **WebSpeed Transaction server** refers to the components (broker, agents, messenger, etc.) that service a WebSpeed request.

- **WebSpeed server** refers to a WebSpeed component of the Progress Developer Studio environment. Derived from the Eclipse Web Tools Platform (WTP), its primary function is to manage the publishing of source code and r-code. The WebSpeed server is configured to be associated with a particular WebSpeed broker. Do not confuse the WebSpeed server with the WebSpeed Transaction server.

- **Web server** refers to the software (Microsoft IIS, Apache, etc.) that responds to requests and delivers content over the Internet.

- **HTTP Server** is an Eclipse WTP server that links your WebSpeed server to a Web server and is also used to publish static content.

- **Publishing** is Eclipse terminology for the ability to compile and copy project resources to an area where you can run, test, and debug them. In the WebSpeed context, publishing is copying dynamic source and r-code to the working directory of a WebSpeed broker, and copying static files to the root directory of a Web server.

- **Modules** are project directories that contain the source code that you intend to publish. You configure modules by specifying the folders within the module that will contain source code. You also create an association between a module and one or more servers. When you publish, the source code is compiled and copied to the associated servers.

---

**Note:** You can view and modify the source folders on the **Modules properties** page. In addition, you can view and modify server/module associations by selecting **Add and Remove** from the context menu of a server in the **Servers** view.

---

**See also**

- [WebSpeed architecture](#) on page 338
- [Web objects](#) on page 340
- [Standard HTML files](#) on page 341
- [Embedded SpeedScript files](#) on page 341
- [CGI Wrapper files](#) on page 342
- [Creating a WebSpeed project](#) on page 342
- [Managing Eclipse WTP servers](#) on page 344
- [Managing the WebSpeed Transaction Server](#) on page 348
- [Publishing Web objects](#) on page 354

---

**Web objects**

WebSpeed applications are composed of one or more Web objects. A Web object is a Progress application file that generates HTML. The WebSpeed agent runs Web objects in response to client requests.

There are several programming methods for creating a source files for Web objects in Progress Developer Studio for OpenEdge:

- Creating a standard HTML file.
• Creating an HTML file that contains Embedded SpeedScript code.
• Creating a CGI Wrapper file.

**Note:** HTML Mapping is another programming method for creating Web objects. HTML Mapping involves mapping form elements in an HTML file to a database field or some other data element. HTML Mapping is not commonly used and is not supported in Progress Developer Studio for OpenEdge. However, you can use the OpenEdge AppBuilder tool if you prefer to use HTML Mapping programming. You can run the AppBuilder from the OpenEdge AppBuilder perspective in Progress Developer Studio for OpenEdge.

In Progress Developer Studio for OpenEdge, you compile standard HTML, HTML with Embedded SpeedScript, or CGI Wrapper files to create executable Web objects.

**See also**
- WebSpeed architecture on page 338
- WebSpeed and Eclipse server terminology on page 340
- Standard HTML files on page 341
- Embedded SpeedScript files on page 341
- CGI Wrapper files on page 342
- Working with Web objects on page 349

**Standard HTML files**

In Progress Developer Studio for OpenEdge, you can compile almost any HTML file into a Web object. The compilation process creates a temporary .w file and then produces an executable .r file. The resulting Web object (that is, the .r file) generates a Web page that is identical to the original HTML file when viewed in a browser.

It might seem unnecessary to convert HTML files into Web objects since HTML files are already viewable in a browser. However, when you run a WebSpeed application, you are running Web objects on a WebSpeed agent. WebSpeed agents cannot process HTML files directly; they can only execute r-code. Therefore, simple or static HTML files must be first converted to r-code so they can be regenerated by a WebSpeed agent.

The Eclipse WTP framework includes a wizard for creating standard HTML files (**File > New > Other > Web > HTML File**).

**See also**
- WebSpeed architecture on page 338
- WebSpeed and Eclipse server terminology on page 340
- Web objects on page 340
- Embedded SpeedScript files on page 341
- CGI Wrapper files on page 342
- Working with Web objects on page 349

**Embedded SpeedScript files**

SpeedScript is a subset of the Progress ABL language. The SpeedScript code is embedded between the `<script Language="SpeedScript">` tag and the `</script>` end tag in an HTML file.
The compilation process creates a temporary .w file and then produces an executable .x file. The r-code recreates the content of the original HTML file and adds the dynamic content or any additional processing logic created by the embedded SpeedScript code.

Progress Developer Studio for OpenEdge includes a wizard for creating embedded SpeedScript files (File > New > SpeedScript). The wizard includes templates that you can use to create a number of standard types of Web pages. In addition, you can customize or add to the standard templates.

See also
WebSpeed architecture on page 338
WebSpeed and Eclipse server terminology on page 340
Web objects on page 340
Standard HTML files on page 341
CGI Wrapper files on page 342
Working with Web objects on page 349

CGI Wrapper files

A CGI Wrapper file is a .w file. The source for CGI Wrapper Web objects contains HTML markup that is "wrapped" by a SpeedScript {&OUT} preprocessor statement. The {&OUT} preprocessor combines with a sub-procedure called process-web-request to generate a valid HTML page. The generated page includes an HTML header that is produced by the CGI Wrapper.

There is no HTML source file associated with a CGI Wrapper Web object. Compilation of a .w source file produces a .x file that is executed by a WebSpeed agent. When executed, CGI Wrapper Web objects dynamically create HTML content that is returned to the client browser.

Progress Developer Studio for OpenEdge includes a wizard for creating CGI Wrapper files (File > New > CGI Wrapper).

See also
WebSpeed architecture on page 338
WebSpeed and Eclipse server terminology on page 340
Web objects on page 340
Embedded SpeedScript files on page 341
Standard HTML files on page 341
Working with Web objects on page 349

Tasks

This section contains information on how to use Progress Developer Studio for OpenEdge to create, test, and deploy WebSpeed applications.

Creating a WebSpeed project

A WebSpeed project is an OpenEdge project specifically tailored for WebSpeed application development. It is usually configured with a WebSpeed server and an HTTP Server, which are servers based on the Eclipse Web Tools Platform. The primary function of these servers is to support publishing your code for running, testing, and deploying your application.
To verify that these servers are running, check the Servers view. The Servers view is one of the default views in the OpenEdge Server perspective. If it is not visible, you can start it by selecting Window > Show View > Other > Server > Servers. In the Servers view, you should see:

- a WebSpeed server associated with a particular WebSpeed broker (for example, wsbroker1 WebSpeed 11.0 at mymachinename)
- an HTTP Server associated with a host that is running a Web server (for example, HTTP Server at localhost)

You can create either or both of these servers, or add additional servers to your workspace before you create your WebSpeed project. See Managing Eclipse WTP servers on page 344 for more information. You can also create the project first and configure the servers afterward.

To create a WebSpeed project:

1. Select File > New > OpenEdge Project from the Progress Developer Studio for OpenEdge main menu bar or the Project Explorer view context (right-click) menu. The New OpenEdge Project wizard opens and displays the Create an OpenEdge Project page on page 105.

2. Type a name in the Project name field.
   The root folder for the new project is created in your current Eclipse workspace, unless you specify a different location.

3. To choose a different location for the project, click and clear the Use default check box. Then either browse to, or enter the path to the desired location.

4. Select Client from the left section.

5. Select the Web Application option button and select the WebSpeed option button.

6. Click Next. The Select AVM and layout options page appears.

7. Choose whether to use a project-specific or a shared AVM.

8. Leave Use TTY for runtime selected.
   By default, WebSpeed projects use TTY for the runtime environment. The only reason you would deselect this option is if you intend to enable a facet for this project that substitutes some other runtime environment.

9. Select the default project layout.
   Leave Use project root directory for source and r-code selected if you intend to store source and r-code in the same folder. Choose the Use separate source and r-code directories option if you want to customize the default file location for source and r-code.

10. Click Next. The Define WebSpeed dynamic content module page appears.

11. Type a name in the Module name field. The default is the project name.
   Modules are project directories that contain the source code that you intend to publish.

12. Type the location of source files in the WebSpeed source folder field.
   The default is a folder named WebSpeed, which will be created as a child of the project folder if you chose Use project root directory for source and r-code, in step 8. If you chose the Use separate source and r-code directories option, the folder will be a child of whatever you specified in the Source directory field.

13. Select the WebSpeed server from the list of supported servers that will be the destination for your published web objects.
    The primary function of these servers is to support publishing your code for running, testing, and deploying your application. The default list contains two servers named:

    wsbroker1 WebSpeed OE_Versionathostname
    wsdynamics WebSpeed OE_Versionathostname
Select wsdynamics only if you are using the Progress Dynamics framework as a development environment. If there are no available servers, you can create them and associate them with the project after you finish creating the project.

14. Type a destination folder for the .w files that are generated when SpeedScript files are compiled. The default is SpeedScriptGen in the project root folder.

15. Click Next. The Define static web content module page appears.

16. Type the name of the destination folder where static HTML pages will be published in the Context root field. The folder will be created as a child of the scripts folder on your web server. The default is to use the same name as the project.

17. Type the name of the folder in the project where static HTML source files will be saved. The default is HTML Content in the project root directory.

18. Select the Web server static from the list of supported servers where static content should be published.

19. Click Next. Specify the PROPATH to use for the project.

20. Click Next. Specify the database connection(s) to use for the project.

21. Click Finish. If the OpenEdge Server perspective is not already open, you will see a message asking if you want to open it. Since the OpenEdge Server perspective supports WebSpeed application development, you should click Yes.

You will see your project and its resources in the project view. Your next step is Configuring WTP servers on page 347.

See also
WebSpeed architecture on page 338
WebSpeed and Eclipse server terminology on page 340
Managing Eclipse WTP servers on page 344
Managing the WebSpeed Transaction Server on page 348
Working with Web objects on page 349
Publishing Web objects on page 354
OpenEdge Server perspective on page 359

Managing Eclipse WTP servers

Progress Developer Studio for OpenEdge uses servers based on the Eclipse WTP (Web Tools Platform) for publishing and running your WebSpeed web objects. The servers that are necessary for WebSpeed application development are the WebSpeed server and the HTTP server.

Progress Developer Studio for OpenEdge automatically creates WebSpeed servers (and servers for other project types) when you start Progress Developer Studio for the first time in a new workspace and create a new WebSpeed project. However, if you change workspaces, you may be required to manually create these servers.

See also
WebSpeed architecture on page 338
WebSpeed and Eclipse server terminology on page 340
Progress OpenEdge Server Monitor on page 359
Server editor for WebSpeed on page 360
Creating a WebSpeed server

You create a WebSpeed server to handle the publishing of your code to a specific WebSpeed broker. The broker can be local or remote. You create multiple WebSpeed servers when you want to publish to a number of brokers and runtime environments. Typically, you would create at least one WebSpeed server for your development environment and another for your deployment environment.

To create a WebSpeed server:

1. From the main menu bar of Progress Developer Studio for OpenEdge, select File > New > Server. The New Server wizard appears.
   An alternative method is to right-click on the Servers view in the OpenEdge Servers perspective and select New > Server from the context menu.

2. On the first page of the New Server wizard, choose OpenEdge WebSpeed from the Select the server type list.

3. If necessary, change the defaults for Server's host name, Server name, and/or Server runtime environment.

   **Note:** By default, the only available runtime environment is the OpenEdge 11 AVM. You can add other available versions of the OpenEdge AVM after clicking Add. Or, you can click Configure runtime environments to add, remove, or edit runtime environments.

4. Click Next. The Define a new WebSpeed broker page opens.

5. Choose an OpenEdge Explorer connection.
   The OpenEdge Explorer connection is necessary for publishing your code to the server. The default connection is Explorer 1 on the local host. You can add, edit, or remove connections by clicking Configure to open the OpenEdge Explorer Connections preference page.

   **Note:** To support publishing remotely, you can configure OpenEdge Explorer connections for remote hosts. To create a remote connection, you must have the host name or IP address, the Web server port (9090 by default), and the login information for the remote machine.

6. Choose the WebSpeed broker that will be the target for your published code. The list of available brokers is based on the resources managed under the OpenEdge Explorer connection that you chose in the previous step.

   **Note:** If you see no available brokers, it is possible that you are not properly logged in to the OpenEdge Explorer connection. This can occur after you have changed the default password (admin) after starting OpenEdge Explorer for the first time. To update the password, click Configure next to the OpenEdge Explorer connection field. Select the connection and click Edit. Update the password and verify the other connection configuration parameters. You can click Test Connection to verify that the connection is functioning.

7. Type a server name.
   The default server name includes the name of the broker, the type of the broker (WebSpeed or AppServer), the runtime environment, and the host. This convention is useful for identifying multiple servers when you are publishing to different machines and different runtime environments. However, you can change the name to suit your own conventions.
8. Click **Next**. The **Define publishing rules** page opens.

9. Select **Use custom publish directory** if you want to change the default publish location in the workspace for a local broker and add the custom location to the **Publish directory** field. This does not change the actual publish directory for the broker itself. It modifies the metadata of the workspace and does not affect users that are accessing the broker remotely. The default publish location is the working directory of the broker, which is specified in a ubroker.properties file. You cannot edit the ubroker.properties file directly from Progress Developer Studio for OpenEdge, but you can specify a publish location that differs from the working directory.

   **Note:** You cannot specify a custom publish directory for a remote broker in the **New Server** wizard. If you chose a remote broker on the previous page of the wizard, the custom publish option will be disabled.

   Use OpenEdge Explorer (OpenEdge > Admin > OpenEdge Explorer) to change a broker's working directory or publish directory. Keep in mind that the publish directory must be included in the PROPATH for the application to compile and run successfully.

   You can also modify the working directory of brokers as a launch configuration setting in the **Run Configurations** dialog. Select Run > Run Configurations from the main menu bar of Progress Developer Studio for OpenEdge. Open the **Startup** tab to modify the working directory.

10. Clear the **Publish source code** and/or **Publish r-code** check boxes if it is preferable to exclude either source or r-code from publishing.

11. Select **Compile on publish** if you want to compile with the broker's AVM rather than the project's AVM.

   **Note:** The **Compile on publish** option is only available if the broker is remote.

12. Click **Next**. The **Define a new Web server** page opens.


   By default, **Web Server URL** is selected. A URL to the WebSpeed broker that you specified on page two of **New Server** wizard is specified.

   **Note:** If you intend to accept the default Web server, be sure to check that the path to the scripts directory on the Web server is correct. The default specifies the Microsoft IIS standard (/Scripts/... on Windows and /cgi-bin/... on Linux).

   An alternative is to select the **Web Server** option and choose from the available Web servers in the workspace. (To add a Web server to the workspace, see **Creating an HTTP server** on page 347.)

14. Click **Next**. The **Add and Remove resources** page opens.

15. From the **Available** list, select the projects that contain the resources that you want to publish.

16. Add the selected projects to the **Configured** list.

17. Click **Finish**.

   To publish the resources that you configured for this server, right-click on the server name in the **Servers** view and select **Publish** from the context menu.

**See also**

on page 338
WebSpeed and Eclipse server terminology on page 340
Creating an HTTP server

An HTTP server manages the publishing of the source and r-code of static files to a directory on the Web server. You can also use the HTTP server as the Web server parameter when you create a WebSpeed server.

To create an HTTP server:

1. From the main menu bar of Progress Developer Studio for OpenEdge, select File > New > Server. The New Server wizard appears.

   An alternative method is to right-click on the Servers view in the OpenEdge Servers perspective and select New > Server from the context menu.

2. On the first page of the wizard, expand the Basic node.

3. Select HTTP Server.

4. If necessary, change the defaults for Server's host name, Server name, and/or Server runtime environment.

5. Click Next. The HTTP Server page opens.

6. If necessary, change the port number, add a URL prefix, or disable publishing (only if there is no static content).

7. Click Next. The Add and Remove page opens.

8. Add any of the available projects that you want to associate with this server to the Configured list.

9. Click Finish. The HTTP server appears in the Servers view.

See also

WebSpeed architecture on page 338
WebSpeed and Eclipse server terminology on page 340
Creating a WebSpeed server on page 345
Configuring WTP servers on page 347
Progress OpenEdge Server Monitor on page 359
Server editor for WebSpeed on page 360

Configuring WTP servers

To configure the WebSpeed server or the HTTP server:

1. Open the Servers view in Progress Developer Studio for OpenEdge.
Note: The **Servers** view is a default view that appears in the console area of the OpenEdge Server perspective. You can start it in any perspective by selecting **Window > Show View > Server** from the main menu bar of Progress Developer Studio.

2. In the **Servers** view, double-click the name of the server that you want to configure. A server editor appears, and it allows you to change all the configuration parameters associated with a particular server.

Note: You can also launch server editors by right-clicking and choosing **Open**, or by pressing **F3**, after selecting a server name in the Servers view.

**See also**

- WebSpeed architecture on page 338
- WebSpeed and Eclipse server terminology on page 340
- Creating a WebSpeed server on page 345
- Creating an HTTP server on page 347
- Progress OpenEdge Server Monitor on page 359
- Server editor for WebSpeed on page 360

**Managing the WebSpeed Transaction Server**

OpenEdge Explorer is the primary tool for managing the WebSpeed Transaction server's brokers and agents. However, you can perform a few basic tasks (adding and trimming) agents from the context menu in the Servers view. In addition, you can perform many configuration tasks from the WebSpeed server's launch configuration. (Select **Run > Run Configurations** from the main menu bar to modify launch configuration settings.)

1. Start OpenEdge Explorer from the main menu bar (**OpenEdge > Admin > OpenEdge Explorer**) of Progress Developer Studio for OpenEdge. In addition, you can start OpenEdge Explorer from the toolbar of the Progress OpenEdge Server Monitor view, or from the context menu of a server in the **Servers** view.

Note: When started, OpenEdge Explorer prompts for an administrator user name and password. The default is **admin** for both. After login, it runs in a browser embedded in the active perspective. For online help, click one of the help buttons in OpenEdge Explorer. If you change the administrator user name or password in OpenEdge Explorer, you must update the respective OpenEdge Explorer connections that exist in your workspace. OpenEdge Explorer connections enable publishing your application to WebSpeed servers.

2. To update OpenEdge Explorer connections:
   a) Select **Window > Preferences** from the main menu bar of Progress Developer Studio.
   b) Select **Progress OpenEdge > Server > OpenEdge Explorer Connections** from the tree view.
   c) Select an existing OpenEdge Explorer connection name and click **Edit**.
   d) Update the **User name** and **Password** fields for each existing connection.

**See also**

- Add/Edit OpenEdge Explorer connections page on page 356
- Progress OpenEdge Server Monitor on page 359
Working with Web objects

This section describes WebSpeed application development using the tools and infrastructure of Progress Developer Studio for OpenEdge.

Adding new Web objects to a WebSpeed project

You use a wizard from the **File > New menu** to add a new Web object source file to a WebSpeed project. Choose one of the following procedures:

- To add a new Embedded SpeedScript source file to a project, select **File > New > SpeedScript** from the main menu bar of Progress Developer Studio for OpenEdge. The **New SpeedScript Wizard** appears and it allows you to choose from a variety of templates. The new file opens in the SpeedScript editor.
- To add a new CGI Wrapper source file to a project, select **File > New > CGI Wrapper** from the main menu bar of Progress Developer Studio for OpenEdge. A dialog where you define the file properties appears. The new file opens in the ABL Editor.
- To add a static HTML source file to a project, select **File > New > Other > Web > HTML File** from the main menu bar of Progress Developer Studio for OpenEdge. The **New HTML File wizard** appears and it allows you to choose from a variety of templates. The new file opens in the SpeedScript editor.

**Note:** Progress Developer Studio for OpenEdge does not support developing Web objects from HTML Mapping source files. However, you can use the OpenEdge AppBuilder tool if you prefer to use HTML Mapping programming. You can run the AppBuilder from the OpenEdge AppBuilder perspective in Progress Developer Studio for OpenEdge.

See also
- Importing Web objects to a WebSpeed project on page 349
- Using the SpeedScript Editor on page 350
- Working with CGI Wrapper files on page 352
- Compiling Web objects on page 353
- Publishing Web objects on page 354
- Debugging Web objects on page 355
- Running Web objects on page 355
- Changing the default Web browser on page 356
- New CGI Wrapper wizard on page 357
- New HTML file wizard on page 358
- New SpeedScript wizard on page 358

Importing Web objects to a WebSpeed project

If you have the source file for an existing Web object, you can import it to a Progress Developer Studio WebSpeed project. Use the **Import Wizard** as follows:

1. Select a WebSpeed project from the **Project Explorer** view.
2. Choose **File > Import** from the main menu bar.
3. When the **Import** wizard opens, select an import source, usually **General/File System**.
4. Click **Next**.

5. In the **From directory** field of the **File system** page, type or browse to the folder that contains the source files.

6. Select the folder and the files that you want to import in the tree view.

7. In the **Into folder** field, type or browse to the destination folder (the default destination is `project_name/WebSpeed`).

8. Review the **Options** section and select any options that are appropriate.

9. Click **Finish**.

10. Find the imported source file in the Project Explorer.

11. Compile the file by right-clicking on the file name and selecting **Progress OpenEdge > Compile**.

---

**Note:** Progress Developer Studio for OpenEdge does not support developing Web objects from HTML Mapping source files. However, you can use the OpenEdge AppBuilder tool if you prefer to use HTML Mapping programming. You can run the AppBuilder from the OpenEdge AppBuilder perspective in Progress Developer Studio for OpenEdge.

---

**See also**

- Adding new Web objects to a WebSpeed project on page 349
- Using the SpeedScript Editor on page 350
- Working with CGI Wrapper files on page 352
- Compiling Web objects on page 353
- Publishing Web objects on page 354
- Debugging Web objects on page 355
- Running Web objects on page 355

**Using the SpeedScript Editor**

In Progress Developer Studio for OpenEdge, you edit embedded SpeedScript files in the OpenEdge SpeedScript Editor. Embedded SpeedScript files are HTML files that contain SpeedScript (a subset of ABL) code contained within HTML `SCRIPT` elements. The OpenEdge SpeedScript Editor supports editing HTML, JavaScript, CSS, as well as SpeedScript.

When you double-click on a file that has an `.htm` or `.html` extension in Progress Developer Studio for OpenEdge, the file opens in the SpeedScript Editor by default.

**Code editing features**

The SpeedScript Editor has many features to assist in writing code. They include content assist, case correction, keyword expansion, hover help and so on. The code editing features that apply to SpeedScript code are identical to ABL editing features. They are more fully documented in the ABL Editor help. Note that SpeedScript editing features usually do not apply to HTML, JavaScript, or CSS elements in a SpeedScript file. For example, case correcting (**Source > Correct case**) only applies to SpeedScript elements.

Most SpeedScript editing features are available from the main context menu, or under the **Source** node of the context menu.

See the topic links below for more information about specific editor features and settings.

**Outline view support**

HTML and SpeedScript sections are shown the **Outline** view when focus is in the SpeedScript Editor.

**WebSpeed markup support**
The SpeedScript Editor also supports these WebSpeed-specific statement and expression escapes:

- `<?WS></?WS>`
- `<!--WSS -->
- `` (back tics)
- `<!--WSE -->`
- `{==}`

See also

Using tabs on page 351
Setting SpeedScript Editor preferences on page 351
Using tabs on page 351
Syntax Checking on page 352

Using tabs

The following tabs appear at the bottom of the SpeedScript Editor view:

- **SpeedScript** -- presents an editable view of the file. It contains editing support for HTML, CSS, JavaScript, and SpeedScript.
- **Generated ABL** -- generates and displays a .w version of the HTML file. By default, the .w file is created in the project folder: ..../SpeedScriptGen/WebSpeed.

**Note:** This tab appears only when you open a file contained in a WebSpeed project. Progress Developer Studio does not support compilation of external files. Also, this tab does not appear if you disable .w file generation on the SpeedScript properties page.

The Generated ABL tab is a read-only view. You can, however, open the generated .w file in the ABL Editor to set breakpoints. If you attempt to edit the file, you will see a warning that you are attempting to edit a derived file.

- **Preview** -- displays a preliminary view of the file that shows how the content will be rendered in a browser. Note that the tab only displays HTML, JavaScript, and CSS content. SpeedScript content is not compiled. In order to run the SpeedScript code, select the file and run it on the server (ALT+SHIFT+X, R).

See also

Setting SpeedScript Editor preferences on page 351
Syntax Checking on page 352

Setting SpeedScript Editor preferences

Some of the SpeedScript editor preferences are set on the same preference pages that apply to the ABL Editor (Window > Preferences > Progress OpenEdge > Editor). Therefore, general preferences that are set for the ABL Editor also apply to the SpeedScript Editor.

However, there are some ABL Editor preferences that are particularly relevant to SpeedScript, including:

- **Color** -- Window > Preferences > Progress OpenEdge > Editor > Colors
  Click on the icon next to SpeedScript to bring up a palette of color choices for SpeedScript elements (`<Script language="SpeedScript"></Script>`) in an HTML file.
Notice the **Color** preference page section, which has links to pages where you can set color preferences for the HTML, CSS, and JavaScript elements in an HTML file.

**File types -- Window > Preferences > Progress OpenEdge > Editor > Build**

By default, the **Compilable file extensions** field specifies `html` and `htm` as file types that can be compiled in Progress Developer Studio for OpenEdge.

Also by default, the SpeedScript extensions field specifies `html` and `htm` as file types that automatically open in the SpeedScript Editor when you double-click on them in the Project Explorer view.

**Templates -- Window > Preferences > Progress OpenEdge > Editor > Templates**

You can edit one of the existing WebSpeed templates (WebSpeed Blank, WebSpeed Frameset, etc.) shown in the list box, or create a new one. The WebSpeed templates listed on this page are available for selection when you use the **New SpeedScript** wizard to create a new SpeedScript file.

There are other editor preference pages available from the tree view of the Preference dialog. To set preferences that apply to anything other than ABL or SpeedScript:

- Expand the **General > Editors > Text Editors** node for Eclipse platform editor settings.
- Expand the **JavaScript > Editor** node for JavaScript settings.
- Expand the **Web > CSS Files > Editor** node for CSS settings.
- Expand the **Web > HTML Files > Editor** node for HTML settings.

**See also**

Using tabs on page 351  
Syntax Checking on page 352

**Syntax Checking**

To check the syntax of the active file in the SpeedScript Editor, right-click and select **Check Syntax** from the context menu.

Syntax errors are also marked ✗ on the vertical rule of the editor after a save.

**Note:** Syntax checking only applies to SpeedScript code. Errors in HTML, CSS, or JavaScript are not reported.

**See also**

Using tabs on page 351  
Opening declarations on page 351

**Working with CGI Wrapper files**

A CGI Wrapper file is an ABL procedure file (`.w`) that includes a standard template for processing web requests. (It does not have an associated HTML source file.) It directs HTML output to the WebSpeed-defined output stream, `WEBSTREAM`, by using `&OUT` statements.

Since it is essentially an ABL procedure file, you use the ABL Editor for editing CGI Wrapper files. The ABL Editor supports editing CGI Wrapper files with basic color coding of HTML markup inside strings, indentation of `&OUT` statements, and content assistance for ABL elements including WebSpeed super procedures.

A wizard is available for creating new CGI Wrapper files (**File > New > CGI Wrapper**).
Compilation of a CGI Wrapper generates r-code that can be executed by a WebSpeed agent.

**Compiling Web objects**

In a WebSpeed project, you can compile SpeedScript (.htm or .html) files or CGI Wrapper (.w) files by:

- Right-clicking on a file name in the Progress Explorer view and selecting Progress > OpenEdge > Compile from the context menu.
- Saving a file open in the editor by typing CTRL+S or selecting File > Save from the main menu bar.

Go to the Build project properties page in order to change the target destination for compiled (.r) files. By default, compiled files are written in the same directory as the source file.

To access the Build properties page, right-click on the project name in the Progress Explorer view. Select Properties > Progress OpenEdge > Build.

When you compile a .htm or .html file, a .w file is created by default. The .w file is necessary for debugging purposes (you cannot debug an HTML file in Progress Developer Studio for OpenEdge).

You can stop the generation of the .w file or change the directory where it is stored on the SpeedScript properties page.

To access the SpeedScript properties page, right-click on the project name in the Progress Explorer view. Select Properties > Progress OpenEdge > SpeedScript.

If your .html file contains WSMETA NAME="wsoptions" and CONTENT="include", an include file (.i file) is created with the same name as the html when you compile the .html file. For example, if you create index.html and hello.html as follows and compile both the .html files, a corresponding hello.i file is created:

**index.html:**

```html
<!DOCTYPE html>
<html>
<head>
<title>Hello World</title>
</head>
<body>
<!--WS

DEF VAR l-hello AS CHAR NO-UNDO.

{ hello.i }
-->
<p>`l-hello`</p>
</body>
</html>
```

**hello.html:**

```html
<!--WSMETA NAME="wsoptions" CONTENT="include" -->
<!--WS

l-hello = "Thanks".
-->
<p>Thanks for including me!</p>
```

**Note:** If you do not see the include file (.i file) in the Progress Explorer view, refresh the project.

**See also**

- Publishing Web objects on page 354
- Debugging Web objects on page 355
- Running Web objects on page 355
Publishing Web objects

You can use the Eclipse publishing functionality to copy files from a WebSpeed project to a local or remote server for testing or deployment. When you publish, you can copy source and r-code to the working directory of a WebSpeed broker, and static files to the root directory of a Web server.

To publish your code:

1. Open the Servers view.
   
   The Servers view is a default view that appears in the console area of the OpenEdge Server perspective. You can start it in any perspective by selecting Window > Show View > Servers from the main menu bar of Progress Developer Studio for OpenEdge.
   
2. Select the WebSpeed server name.
   
   Next to the server name, there is a status label that indicates the state of the server (Stopped, Starting, or Started). In addition the label indicates if files are current (Synchronized) or if they need to be published again (Republish).
   
   When a WebSpeed server is configured with one or more projects in the workspace, it becomes an expandable node with each project module listed under it. Each project will have a label which indicates if the files are in sync with the project or if they need to be published again.
   
3. Select Publish in the context menu.
   
   The publish target is specified in the server's configuration, which you can view and modify in the Server editor by double-clicking on the server name.
   
   When you publish, all the project modules that need updating are published.

See also

Managing Eclipse WTP servers on page 344
Compiling Web objects on page 353
Debugging Web objects on page 355
Running Web objects on page 355
Progress OpenEdge Server Monitor on page 359
Server editor for WebSpeed on page 360

Publishing Web objects to a remote server

To configure publishing to a remote server:

1. From the management console list frame for the remote machine (http://remote-machine-name:9090), select Resources > remote-container-name > OpenEdge > WebSpeed.

2. Select the WebSpeed broker you want to publish to.

3. In the detail pane, click Configuration.

4. From the Configuration page, click the Broker tab.

5. In the Advanced Features section of the page, verify that the Publish directory property is set to a valid location. If the property value says Not set, click Edit at the top of the page.

6. Type the full path to the publish directory in the property field.

7. Click Save.
8. On the local machine, create an OpenEdge Explorer connection for the remote server as follows:
   a) Select Preferences > Progress OpenEdge > Server > OpenEdge Explorer.
   b) Click Add. The Add OpenEdge Explorer Connection dialog opens.
   c) Type the connection name.
   d) Type the Host name or IP address.
   e) Specify the port. (The default port is 9090.)
   f) Type the user name. (The default user name is admin.)
   g) Type the password. (The default password is admin.)
   h) Provide the OpenEdge Explorer URL. (If you are using the default host name and port number, the URL is http://localhost:9090.)
   i) Click Create Default Server and Finish. If there are no default servers available in the Servers view, a default server will be created and will appear in the Servers view.

9. From the Server Editor view, review the Publish directory setting to verify that the correct path is shown.

**Debugging Web objects**

Progress Developer Studio for OpenEdge has full support for debugging Web objects.

**Note:** A .w file is necessary in order to set at-line breakpoints. Since CGI Wrapper files are .w files by definition, you can set breakpoints directly in the source file that you are editing.

However Embedded SpeedScript files are HTML files, so you must open a .w file generated from the HTML file before you can set an at-line breakpoint. The default SpeedScript project property setting is to automatically generate a .w file in the SpeedScriptGen folder. If you have the HTML file open in the SpeedScript editor, you can switch to the Generated ABL tab to view the .w file and set breakpoints. Alternatively, you can open the .w file by double-clicking on it in the Project Explorer view.

To set an at-line breakpoint:

1. Click on the line where you want to set the breakpoint.
2. Right-click and select Toggle Breakpoint from the context menu.

When you insert an at-line breakpoint, a marker appears in the left margin.

**Note:** For more information about WebSpeed debugging, select a topic from the list below.

**Running Web objects**

To run one of the Web objects in a WebSpeed project:

1. Verify that the server is running in the Servers view.
   The label next to the server name should be Started, Synchronized. Right-click on the server name and choose Start or Publish if necessary.

2. Right-click on a source file.
3. Select Run As > Run on Server.
Note: The **Run On Server** wizard appears where you can choose an existing server or define a new one.

4. Choose a server and click **Next**.

5. On the **Add and Remove** page, verify that the module that contains your source file is on the Configured list.

6. Click **Finish**.

   The user interface is displayed in an internal browser that appears in your workspace. Any other output is directed to a separate TTY window.

You should also be aware of two other methods for running your code:

- You can click the menu button next to the **Run As** button on the toolbar. The menu allows you to select an available server without going through the **Run on Server** wizard.

- You can enter a URL of the web object in an external browser. This is particularly useful if you want to see how your application is rendered in alternative Web browsers. In the URL, you specify the path to the WebSpeed broker followed by the name of the web object. For example:

  `http://localhost/cgi-bin/cgiip.exe/WService=wsbroker1/MyWebObject`

---

**Changing the default Web browser**

To change the Web browser that Eclipse launches when you run a WebSpeed Web object:

1. Select **Window > Preferences** from the main menu bar of Progress Developer Studio for OpenEdge.
2. Expand the **General** node of the tree view.
3. Select **Web Browser**.
4. Select **Use external web browser** if you want to use a browser other than the Eclipse internal web browser.
5. Choose a preferred browser from the list of installed browsers or add a new browser.

---

**Reference**

**Add/Edit OpenEdge Explorer connections page**

By default, an OpenEdge Explorer Connection profile is created when a new workspace is launched. You can manually edit the connection or create a new connection from the **Add/Edit OpenEdge Explorer connections** page.

From the main menu bar of Progress Developer Studio for OpenEdge, select **Window > Preferences > Progress OpenEdge > Server > OpenEdge Explorer Connections** to access the main preference page. Click **Add** or **Edit** to start the **Connection Profile** page, which contains the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Connection name</strong></td>
<td>Specify a unique connection name. The name for the default connection is <strong>Explorer 1</strong>.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Host name/ IP address</td>
<td>Specify local or remote host name or IP address</td>
</tr>
<tr>
<td>Port</td>
<td>Specify the OpenEdge Explorer port number. The default is 9090</td>
</tr>
<tr>
<td>User name</td>
<td>Specify a user name. The default is admin.</td>
</tr>
<tr>
<td>Password</td>
<td>Specify a password. The default is admin. However, you will need to update this after you start OpenEdge Explorer for the first time.</td>
</tr>
<tr>
<td>OpenEdge Explorer URL</td>
<td>Shows the URL that you can enter in a browser to access OpenEdge Explorer.</td>
</tr>
<tr>
<td>Use secure connection</td>
<td>Select to use a secure HTTPS connection instead of the default HTTP connection.</td>
</tr>
</tbody>
</table>

### See also
- Creating a WebSpeed project on page 342
- Managing Eclipse WTP servers on page 344
- Publishing Web objects on page 354

## New CGI Wrapper wizard

To add a new CGI Wrapper file to your WebSpeed project, select **File > New > CGI Wrapper** from the main menu bar of Progress Developer Studio for OpenEdge. The **New CGI Wrapper** wizard appears.

The **New CGI Wrapper** wizard consists of a single page, Create a new CGI wrapper procedure, which contains the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Container</td>
<td>Specify the folder that will contain the new CGI wrapper file. To ensure that the file will be published, specify the WebSpeed Module folder. The default is <em>project_name</em>/Webspeed.</td>
</tr>
<tr>
<td>File name</td>
<td>Specify a unique name for the new CGI wrapper file.</td>
</tr>
<tr>
<td>Description</td>
<td><em>(Optional)</em> Type a description of the of the new CGI wrapper file.</td>
</tr>
<tr>
<td>Purpose</td>
<td><em>(Optional)</em> Type explain the purpose of the new CGI wrapper file.</td>
</tr>
<tr>
<td>Author</td>
<td><em>(Optional)</em> Type the name of the creator of the new CGI wrapper file. By default, this field contains your user name.</td>
</tr>
<tr>
<td>Add AppBuilder markup</td>
<td>Choose if you intend to open the new CGI wrapper file in the AppBuilder (for example, to use ADM building blocks for UI development). When this option is selected you can open the new CGI wrapper file in either the Progress Developer Studio for OpenEdge ABL editor or the AppBuilder.</td>
</tr>
</tbody>
</table>
New HTML file wizard

To add a new static HTML file to your WebSpeed project, select File > New > Other > Web > HTML File from the main menu bar of Progress Developer Studio for OpenEdge. The New HTML File wizard appears.

The first page of the wizard, Create a new HTML file, contains the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter or select the parent folder</td>
<td>Type the path of the folder that will contain the new file, beginning with the name of the project folder. Or, select the folder from the tree view.</td>
</tr>
<tr>
<td>File name</td>
<td>Type a unique name for the SpeedScript file.</td>
</tr>
<tr>
<td>Advanced</td>
<td>Expand to find and select an existing file from the file system.</td>
</tr>
</tbody>
</table>

The second and final page of the wizard, Select HTML Template, contains the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use HTML Template</td>
<td>Select if you want to choose one of the available HTML templates from the Template list. When you choose a template, its HTML markup appears in the Preview pane.</td>
</tr>
<tr>
<td>HTML Templates</td>
<td>Select if you want to add, edit, or remove a template. Clicking the link takes you to the HTML editor Templates preference page where you can add, remove, edit, import or export templates.</td>
</tr>
</tbody>
</table>

New SpeedScript wizard

To add a new embedded SpeedScript file to your WebSpeed project, select File > New > SpeedScript from the main menu of Progress Developer Studio for OpenEdge. The New SpeedScript wizard appears.

The first page of the wizard, Create a new SpeedScript file, contains the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter or select the parent folder</td>
<td>Type the path of the folder that will contain the new file, beginning with the name of the project folder. Or, select the folder from the tree view.</td>
</tr>
<tr>
<td>File name</td>
<td>Type a unique name for the SpeedScript file.</td>
</tr>
<tr>
<td>Advanced</td>
<td>Expand to find and select an existing file from the file system.</td>
</tr>
</tbody>
</table>
The second and final page of the wizard, **Select SpeedScript Template**, contains the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use WebSpeed Template</td>
<td>Select if you want to choose one of the available WebSpeed HTML templates from the list below. When you choose a template, its HTML markup appears in the Preview pane.</td>
</tr>
<tr>
<td>OpenEdge Templates</td>
<td>Select if you want to add, edit, or remove a template. Clicking the link takes you to the OpenEdge Editor Templates preference page, where you can add, remove, edit, import or export templates.</td>
</tr>
</tbody>
</table>

### OpenEdge Server perspective

The OpenEdge Server perspective is tailored for WebSpeed (and AppServer) application development. When you create a new WebSpeed project, the project wizard allows you to start the OpenEdge Server perspective on completion. You can also start it from the main menu bar (**Window > Open Perspective > OpenEdge Server**).

The following are the default views in the OpenEdge Server perspective:

- Project Explorer
- Outline
- DB Structure
- Properties
- Console
- Problems
- Tasks
- Servers
- Progress OpenEdge Server Monitor

In addition, the OpenEdge Server perspective is designed to open static HTML, SpeedScript, and CGI Wrapper files in an appropriate editor.

**See also**

[Creating a WebSpeed project on page 342](#)

### Progress OpenEdge Server Monitor

You use the Progress OpenEdge Server Monitor to view the status and properties of the WebSpeed broker. It is a default view that appears in the console area of the OpenEdge Server perspective. You can start it in any perspective by selecting **Window > Show View > Progress OpenEdge Server Monitor** from the main menu bar of Progress Developer Studio for OpenEdge.
You cannot manage WebSpeed brokers and agents from the Progress OpenEdge Server Monitor. The OpenEdge Explorer (OpenEdge > Admin > OpenEdge Explorer) is the primary tool for managing the WebSpeed Transaction server. However, you can trim and add agents from the context menu in the Servers view from the Progress OpenEdge Server Monitor view’s toolbar. (Note that the buttons are disabled if the server has not started.)

See also
Managing Eclipse WTP servers on page 344
Managing the WebSpeed Transaction Server on page 348

Server editor for WebSpeed

You can launch a server editor by double-clicking a server name in the Servers view.

The following configuration options are available when you choose a WebSpeed server in the Server view:

<table>
<thead>
<tr>
<th>Section</th>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Information</td>
<td>Server Name</td>
<td>Specify the name to assign to the server and to display in the Servers view. The default name includes the OpenEdge version, the server type and the machine name.</td>
</tr>
<tr>
<td></td>
<td>Host Name</td>
<td>Specify localhost, the fully qualified DNS name, or the IP address of the host machine where the server is running. Note that this value is automatically determined when you choose a broker in the Broker name field.</td>
</tr>
<tr>
<td>Runtime Environment</td>
<td></td>
<td>Specify the name of the server runtime environment as defined in the Preferences page (Window &gt; Preferences &gt; Server &gt; Runtime Environments). The server runtime environments define the runtime environment of an application server for compiling your application. To change the server runtime environment, click Runtime Environment.</td>
</tr>
<tr>
<td>Open launch configuration</td>
<td></td>
<td>Open the Edit Configuration dialog to check or change configuration settings for starting the server.</td>
</tr>
<tr>
<td>Connection</td>
<td>OpenEdge Explorer connection</td>
<td>Select a connection to OpenEdge Explorer. Click Configure to modify the existing connection or to add a new connection.</td>
</tr>
<tr>
<td></td>
<td>Broker name</td>
<td>Select an available broker. To create a new broker, use OpenEdge Explorer. You cannot create a new broker in Progress Developer Studio for OpenEdge.</td>
</tr>
<tr>
<td>Section</td>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Publishing</td>
<td>Never publish automatically</td>
<td>Select an automatic publishing option.</td>
</tr>
<tr>
<td></td>
<td>Automatically publish when resources change</td>
<td>You can choose to publish after a resource change (that is, after a file save event), after a build event, or you can disable automatic publishing. Publishing occurs after the time specified in <strong>Publishing interval</strong> elapses.</td>
</tr>
<tr>
<td></td>
<td>Automatically publish after a build event</td>
<td></td>
</tr>
<tr>
<td>Timeouts</td>
<td>StartStop</td>
<td>Specify time limit for server operations</td>
</tr>
<tr>
<td>Publish Location</td>
<td>Use custom publish directory</td>
<td>Select to set a custom publish directory rather than publishing to the server working directory. If you choose to set the custom directory, type its location in the <strong>Publish directory</strong> field.</td>
</tr>
<tr>
<td></td>
<td>Publish source code</td>
<td>Choose to publish source code. This option is selected by default.</td>
</tr>
<tr>
<td></td>
<td>Publish r-code</td>
<td>Choose to publish r-code. This option is selected by default.</td>
</tr>
<tr>
<td></td>
<td>Compile on publish</td>
<td>Choose to compile code when you publish to a remote server. (Note that you can select this option only if the <strong>Publish r-code</strong> option is not selected.)</td>
</tr>
<tr>
<td></td>
<td>Compile options</td>
<td>Add any compile options, such as startup or other parameters. Specify each option with a space between its name and value, and also insert a space between each option.</td>
</tr>
<tr>
<td>Web Server URL</td>
<td>Web Server URL</td>
<td>Select to use the default URL associated with the WebSpeed broker. Note that you may need to edit the URL if the scripts folder of the Web server differs from the default: <code>host_name/Scripts</code>. For example, if you are using an Apache Web Server, the scripts folder is commonly: <code>host_name/cgi-bin</code>.</td>
</tr>
<tr>
<td></td>
<td>Web Server</td>
<td>Select to use a Web server other than the default. If there are no other Web servers available from the drop-down list, click <strong>Add</strong>. Then select <strong>Basic &gt; HTTP Server</strong> to launch the <strong>New HTTP Server</strong> wizard.</td>
</tr>
<tr>
<td></td>
<td>CGIIP URL</td>
<td>Specify the folder on the Web server that contains the WebSpeed Messenger.</td>
</tr>
<tr>
<td></td>
<td>URL Preview</td>
<td>Shows the URL, based on the content of the <strong>Web Server</strong> and <strong>CGIIP URL</strong> fields, of the WebSpeed broker.</td>
</tr>
</tbody>
</table>

**See also**

[Managing Eclipse WTP servers](page344)
WebSpeed Workshop and WebTools

WebSpeed Workshop is a browser-based, server-side utility that consists of a set of tools called WebSpeed WebTools.

You can access WebSpeed Workshop directly from Progress Developer Studio for OpenEdge when you are in the OpenEdge Server perspective. Right-click on a WebSpeed Server in the Server view and select Launch WebSpeed Workshop from the context menu. You will see a menu of the available WebTools in a browser that appears in the console area of the perspective.

You can also access WebSpeed Workshop from an external browser by specifying the Workshop URL, using the following format:

\[host_name/script_dir/messenger/WService=wabroker1/workshop\]

where messenger can be cgiip.exe, wsisa.dll, wsasp.dll, or wsnsa.dll depending on what Web server you are using.

WebSpeed Workshop has its own help system. You can also learn more about the individual WebTools utilities in OpenEdge Getting Started: WebSpeed Essentials in the OpenEdge Product documentation.
Introducing the ABL Editor

The ABL Editor is customized for working with ABL code. It supports many features to make coding easier, including:

- Code-generation wizards for creating new files and adding code blocks to existing files
- Customizable code-completion assistance and syntax help
- Macros
- Auto-formatting
- Color-coding of syntax elements to improve readability
- Drag-and-drop capability to add schema references to code
- Integration with the OpenEdge Visual Designer (Windows only)
- Support for editing CGI Wrapper files for WebSpeed application development

By default, an ABL Editor buffer opens whenever you create an ABL source file or open an existing one (unless the file type is associated with a specialized OpenEdge editor such as the AppBuilder or the Visual Designer). You can have multiple buffers open at the same time, including ABL Editor buffers and other editors.

Preferences

You can control many aspects of the ABL Editor's behavior through preference settings. The available options are in two categories, OpenEdge-specific preferences and standard Eclipse preferences.

OpenEdge-specific preferences

Preferences that apply exclusively to the ABL Editor include options for editing assistance, build behavior, color-coding of syntax elements, and the use of annotations and template (macros).
To view or set these options, select Window > Preferences > Progress OpenEdge > Editor

**Eclipse preferences**

Like all text editors that run in the Eclipse framework, the ABL Editor inherits the Eclipse editor preferences. To view or set these options, select Window > Preferences > General > Editors

---

**Note:** For information on the Eclipse text editor preferences, see the *Workbench User Guide*, starting with Editors.

---

For details, see the following topics:

- Concepts
- Tasks
- Reference

---

**Concepts**

**ABL Resources in Progress Developer Studio for OpenEdge**

Progress Developer Studio for OpenEdge provides support for several specialized types of projects and the different ABL source file types that these projects contain. Although you use the ABL Editor when working with all such source files, the nature of the applications you develop affects the way you create projects, files, and content. This section briefly describe the principal types of ABL code and some guidelines for working with them.

**See also**

[Introducing the ABL Editor](#) on page 363

**Procedural code**

Many different types of OpenEdge projects and applications include ABL persistent procedures. For the purpose of this discussion, however, procedural code refers to non-specialized procedure files (usually named with a .p extension) and include files (usually named with a .i extension). In Progress Developer Studio for OpenEdge, these resources are referred to as:

- ABL Procedure
- ABL Include

The procedural programming model differs significantly from the object-oriented (class-based) model, but objects of both types can communicate with each other and can coexist in the same application. In general, the run-time nature of procedural coding supports a dynamic coding model more easily than does class-based programming. Class-based programming, on the other hand, supports a simplified program structure that can be easier to maintain and can facilitate reuse of code among objects. Because you can mix procedures and class-based objects in the same application, you can choose the model that best serves a given programming task in your application.
Object-oriented code

ABL originated as a procedure-based language. In later releases, extensions were added that allow you to use an object-oriented development model, in which the principal structure is a class rather than a persistent procedure. The object-oriented model includes support for such features as polymorphism, delegation, interfaces, and overloading. Basic object-oriented resources in Progress Developer Studio for OpenEdge are:

- ABL Class
- ABL Interface

Note: GUI for .NET applications use object-oriented ABL code, but Progress Developer Studio for OpenEdge provides specialized tools and menu options for programming them. Therefore, this discussion treats GUI for .NET code as a separate category from basic object-oriented code.

Class-based objects can coexist in the same application and communicate with procedure-based objects (persistent procedures). You can use most ABL syntax within classes, and for the most part, you can use it in just the same way as in procedural code.

Class-based programming supports a simplified program structure that can be easier to maintain than procedural code and can facilitate reuse of code among objects. The run-time nature of procedural coding, on the other hand, generally supports a dynamic coding model more easily than does class-based programming. Because you can mix procedures and class-based objects in the same application, you can choose the model that best serves a given programming task in your application.

For more information on object-oriented extensions, see OpenEdge Getting Started: Object-oriented Programming in the Product Documentation section of the Progress Software Developer’s Network Web site.

See also
Procedural code on page 364
GUI for .NET code on page 365
Writing object-oriented code on page 385

GUI for .NET code (Windows only)

GUI for .NET refers to ABL applications that have a graphical user interface built with .NET forms and controls. To develop such applications in Progress Developer Studio for OpenEdge, you use the Visual Designer, which provides an accurate visualization of the GUI as you design it and greatly simplifies the coding process by automating much of the code generation. The Visual Designer is integrated with the ABL Editor, giving you easy access to both the graphical design tools and the ABL source code.
GUI for .NET resources in Progress Developer Studio for OpenEdge are:

- ABL Form
- ABL Dialog
- ABL MDI Form
- ABL Inherited Control
- ABL User Control

These resources are class files that have the same characteristics and requirements as basic classes. When you create a GUI for .NET resource in Progress Developer Studio for OpenEdge, though, it opens in the Visual Designer. By default, when you edit an existing GUI for .NET class, it also opens in the Visual Designer, although you can explicitly choose to open it in the ABL Editor instead.

The Visual Designer subsection of the Progress Developer Studio for OpenEdge online help explains how to use the tools.

See also
Object-oriented code on page 365
Writing object-oriented code on page 385

AppBuilder code

Progress Developer Studio for OpenEdge includes, on Windows only, an OpenEdge AppBuilder perspective, which allows you to run the AppBuilder embedded in an Eclipse view. When you do so, you have access to the AppBuilder design tools, but you use the ABL Editor (rather than the Section Editor) to edit the source code.

AppBuilder resources in Progress Developer Studio for OpenEdge are:

- ABL Structured Procedure
- ABL Structured Include

When you create such resources, Progress Developer Studio for OpenEdge automatically generates default AppBuilder markup. These resource types are supported on non-Windows platforms, even though the AppBuilder itself runs only on Windows.

You have access to the built-in AppBuilder online help while using the ADM tools in Progress Developer Studio for OpenEdge.

AppServer and WebSpeed code

For n-tier application development in Progress Developer Studio for OpenEdge, you can define and monitor AppServer connections, define AppServer code modules and publish their content to the appropriate servers, add and trim agents, and stop and start brokers.

Progress Developer Studio for OpenEdge supports WebSpeed application development with the OpenEdge SpeedScript Editor, as well as with the integration of the OpenEdge AppBuilder.

AppServer code modules can include any valid ABL source files, so no resource types are specifically designated as AppServer resources. WebSpeed resources in Progress Developer Studio for OpenEdge are:

- CGI Wrapper
- SpeedScript
The OpenEdge Server perspective includes the ABL Editor and several views that support common server-related tasks.

**Excluded resources**

In a Progress Developer Studio for OpenEdge project, you can exclude or include compilable resources (like p, cls, or htm) from your project build. **Exclude from Build** excludes the selected resources from the build and **Include in Build** includes the resources that were excluded from the build previously. You can access these options from the context menu of the resource or from the **Source** tab of the **Build** properties page.

**Note:** See Editor Colors options on page 439 and Editor Build options on page 438 for the list of compilable file extensions defined for your project.

The excluded resources entries are saved to the **Excluded** node on the **Source** tab, and only the entries that are present under the **Source** tab are considered for building. The Progress Developer Studio for OpenEdge compiler checks the exclusion entries before compiling a file. Once a file is excluded, then the r-code of that file is deleted. If the earlier excluded file is now included, then the respective file is compiled and the corresponding r-code for it is created. The project is compiled (r-code is generated) only when the relative path of the project being included does not match with any other resource relative path, which is at the top of order on **Source** tab.

For new projects created in the Progress Developer Studio for OpenEdge 11.0, the project **ROOT** node is explicitly added to the **Source** tab by default. A new node **Excluded** is added to the project **ROOT** on the **Source** tab. In releases prior to 11.0, the **Source** tab contained no entries by default even though the project root is included in the build implicitly. When an older version project is opened with Progress Developer Studio for OpenEdge 11.0, note the following:

- If the project does not contain any directories under the **Source** tab, the project **ROOT** node is added under the **Source** tab by default. A warning message appears in the **Problems** view as **Project root is added under the source tab. Only the entries under the source tab are built.**

- If the project contains folders under the source tab, no project **ROOT** is added. A warning appears in the **Problems** view as **only the entries under the Source tab are built.**

**Note:** The warning message displays only the first time a project created in a release prior to 11.0 is opened with 11.0. The warning message is removed from the **Problems** view, when you add or modify entries in the **Source** tab of the **Build properties** page.

When you remove the **ROOT** node or last existing node from the **Source** tab, a warning message appears as **Removing all entries from the Source tab results in no resources being built. Do you want to continue?**. Only the entries that appear under **ROOT** node on the **Source** tab gets built automatically. However, you can manually compile the files in your project.

**See also**

- Excluding resources from the build on page 418
- Including resources in the build on page 420
- Building resources using working sets on page 421
- Compiling, testing, and running ABL programs on page 414
ABL Editor Features

This section is a summary of the main features that the ABL Editor offers to help you increase productivity and efficiency when developing ABL applications.

See also
Introducing the ABL Editor on page 363

Formatting and display features

The ABL Editor offers several features that help you format your ABL code for readability and consistency. These features include:

- **Code folding** - Code folding temporarily hides sections of your code so that you can more easily scan the broader structure. Examples of code sections that you can fold are include files, internal procedures, functions, and methods.
- **Color coding** - The ABL Editor displays different syntax elements in different colors to make them easy to recognize at a glance. You can use the default color scheme or go to the ABL Editor Preferences page to create your own.
- **Keyword casing** - You can elect to have the ABL Editor display syntax keywords in all uppercase or all lowercase characters, or just as you type them. Your casing preference can be applied as you type, on demand, or when you save the file.
- **Keyword expansion** - The ABL Editor can expand abbreviated keywords to their fully spelled equivalents (for example, changing DEF to DEFINE), either as you type or on demand.
- **Indentation and alignment** - The ABL Editor can help you improve readability by automatically indenting lines as you type, when you paste text, or on demand. A separate option (tabular formatting) lets you automatically left-align like elements of statements within a code block, further improving readability.

See also
Formatting and displaying code on page 396
ABL Editor preferences on page 433
ABL Editor menus on page 446

Navigation features

Available tools and techniques for navigating easily through ABL source files include:

- **Outline view** - This view, part of the default OpenEdge Editor perspective, shows all structural elements of the current file, grouped by category. Clicking on a node in the outline takes you quickly to the corresponding location in the code.
- **Quick outline** - Similar to the Outline view, Quick Outline appears on demand in a pop-up window and offers several options for viewing and navigating the code structure.
- **Opening included files** - The context (right-click) menu in the ABL Editor lets you open an include file declared in the current file. You can also open include files from the Outline view and Quick Outline.
- **Preprocessor view** - This view displays code obtained after all the preprocessors (include files, preprocessor definitions, and logic) are expanded. Syntax coloring and highlighting are applied to the code in the Preprocessor view, so it appears the same as the code in the ABL Editor.
See also
Navigating on page 399

Code assistance
The ABL Editor offers several forms of assistance to facilitate writing ABL code. Assistance options are grouped into two categories: context assistance and code-completion assistance.

Context assistance includes the following options:

- **Help as you type** - When you type a keyword followed by a space, a pop-up window shows syntax that is valid to complete the line you are typing.
- **Bracket-matching** - A marker indicates the paired element, such as a bracket or a keyword that begins or ends a code block, corresponding to the element at the current cursor position.
- **Hover help** - When you stop the cursor over an ABL statement, a database field reference, or a system handle, a pop-up window shows relevant syntax help.

Code completion assistance reduces errors and saves typing by automatically entering text. It includes the following options:

- **Syntax completion** - When you press CTRL+SPACE, the ABL Editor shows a list of syntax elements that are valid for the context. You can select a proposal from the list to enter it automatically.
- **Auto-bracketing** - When you type the first of two paired elements, such as an opening parenthesis or bracket, the corresponding closing element is automatically entered.
- **Preprocessor proposals** - Displays proposals for built-in or defined preprocessors. The value of the preprocessors at the selected offset is used to evaluate the proposals.

You set your preferences for these assistance options at the Editor Preferences page.

See also
Using assistance features on page 389
Working with preprocessor elements on page 403
Editor Assistance options on page 436

Code annotations
ABL annotations enable you to include meta data in your source code. Among the many uses for ABL annotations are identifying procedures and user-defined functions for use with the Progress Sonic Enterprise Service Bus (ESB), and indexing your code base with the OpenEdge Meta Catalog. The compiler treats annotations as comments, so they do not affect how your code runs.

The ABL Editor provides a wizard that simplifies the task of adding annotations to selected files. The wizard provides templates for ESB and Meta Catalog annotations. You can also add your own custom annotation templates.

See also
Using annotations on page 406
Editor Annotations options on page 435
Code parsing

Many features of the ABL Editor depend on the availability of an accurate in-memory representation of the structure of the ABL code that you are editing. Progress Developer Studio for OpenEdge includes a parser that continuously and automatically analyzes (parses) the code as you edit it. The parser updates the in-memory representation of the code structure. The in-memory representation of the code enables code-completion assistance, an accurate outline in the Outline view, color-coding in the ABL Editor, and many other Progress Developer Studio for OpenEdge features.

The parser performs a new analysis and updates its in-memory representation whenever a change to the code is followed by a specified interval (by default, 300-milliseconds) during which no editing activity occurs. In other words, while you type, delete, or move text in an ABL source file, the parser waits for a pause. As soon as the 300-millisecond interval elapses with no changes, the parser performs an update.

If you are working on large files, you might find that performance improves when you increase the default 300-millisecond interval. You can adjust the interval for re-parsing on the Editor Assistance preference page.

From the main menu of Progress Developer Studio for OpenEdge, go to Window > Preferences > Progress OpenEdge > Editor > Assistance.

See also
Using code-completion features on page 393
Adding SQL connection profiles on page 436

Support for structured error handling

Many Progress Developer Studio for OpenEdge file-creation and code-generation tools provide options to include ABL code for structured error-handling features. These options, available in the appropriate wizards and menus, are:

- Include the routine-level error-handling statement `ROUTINE-LEVEL ON ERROR UNDO, THROW` when creating new classes and procedures. This option is selectable in the file-creation wizard.

- Include a `CATCH` block and/or a `FINALLY` block when adding a new function, internal procedure, or method. These options are selectable in the code-generation wizard.

- Add a `CATCH` block and/or a `FINALLY` block surrounding selected code in an existing ABL source file. These commands are available on the Source > Surround With. The blocks are also included in code-assistance completion proposals where applicable.

*Note:* For detailed information on these ABL features, refer to *OpenEdge Development: Error Handling* and *OpenEdge Development: ABL Reference* available in the Product Documentation category on PSDN.

See also
Creating ABL source files on page 381
Writing procedural code on page 383
Writing object-oriented code on page 385
Code-generation wizards on page 450
Finding references

Progress Developer Studio for OpenEdge helps you find all occurrences of a specific ABL construct in an ABL file, project, or workspace. It enables you find declarations, references, and occurrences of procedural and object-oriented ABL constructs such as procedures, functions, methods, fields, and variables through editors, menu items, and views.

The references to the following are not found:

- System level members for example, Integer function
- Files that reside on network file paths
- Constructs in r-code
- Constructs in file types other than .i, .w, .cls, or .p file, for example, spring.xml or JSON file
- Constructs in speedscript files
- Dynamic instances such as dynamic casts that use expression values, expressions used as parameters, statements that include pre-processors, and handles.

PDSOE lets you find references to the following ABL constructs:

- **Include file**
  You can find references to Include file (.i), Procedure file (.p), or class in an Include file in a project or a workspace. The search depends on PROPATH; any path specified in procedure or function is relative to the order of entries of the relevant resource in the PROPATH project. If the path in an Include statement resolves to two resources in PROPATH, the reference to the entry that is listed first in the order is displayed. However, the references are not found if the Include statement has a pre-processor value instead of an include file name in text format, for example, as follows:

  &SCOPED-DEFINE INCLUDE_FILE_NAME custom.i

  /* Include statement with file name */
  {custom.i}

  /* Include statement with pre-processor value */
  {{&INCLUDE_FILE_NAME}}

- **Internal procedure or user defined function**
  You can find references to an internal procedure or a user-defined function (UDF) defined in a Procedure (.p) file or an Include (.i) file. All the references to the internal procedures in the RUN statement and the function calls in the same file are found. However, the references in the other files that invoke the internal procedure using handle variables and references to a UDF that is invoked as a dynamic function using the DYNAMIC-FUNCTION statement are not found. The reference to Procedure (.p) file included as a pre-processor directive is also not found.
The following example displays the RUN statement with a procedure name and expression:

```abl
define variable procname as character.
procname = "myprocedure".
/* run with direct internal procedure name */
run myprocedure.
/* run with expression */
run value(procName).
```

### Internal Procedures

```abl
procedure myprocedure:
    message "Inside Custom.myProcedure"
    view-as alert-box.
end procedure.
```

The following example shows you how to create a handle:

```abl
/* HandleTest.p: Invoking an internal procedure using handle*/
define variable h as handle no-undo.
run zsource.p persistent set h.
run intProc in h.
```

The following example shows you how to create Source.p for the above handle:

```abl
/* Source.p : Procedure file with an internal procedure */
procedure intProc:
    message "in zSource.p>intProc" view-as alert-box.
end procedure.
```

The following example displays the DYNAMIC-FUNCTION statement:

```abl
/* h-FuncProc.p -- contains CtoF and other possible useful functions. */
function CtoF returns decimal (input dCelsius as decimal):
    return (dCelsius * 1.8) + 32.
end function.
```

```abl
/* Custom.p */
define variable dTemp as decimal no-undo.
define variable hFuncProc as handle no-undo.
define variable func-name as character no-undo init "CtoF".
repeat dTemp = 0 to 5:
    display dTemp label "Celsius"
    /* Invoking function defined in other procedure dynamically */
    dynamic-function (func-name in hFuncProc, dTemp) label "Fahrenheit"
    with frame f 10 down.
end.
```

```abl
delete procedure hFuncProc.
```

**External procedure**

You can find references to an external procedure in a project or a workspace, the functionality evaluates all the ABL resources and invokes the RUN statement. However, if the RUN statement uses an expression to evaluate the external procedure name dynamically, the references are not found.
The following is an example of the RUN statement with an external procedure name, and path and expression:

```plaintext
DEFINE VARIABLE procName AS CHARACTER.
procName = "custom".
/* Run using external procedure name */
RUN custom.p.
/* Run with expression value */
RUN value(procName + ".p")
```

- **Variable**

You can find references to a variable defined in a procedure or class in the working file. Finding references to SHARED and GLOBAL variables are limited. A SHARED variable is created in a source procedure and accessed by all other procedures. There can be multiple source procedure files for a single consumer procedure, so the value of the variable depends on the source being executed and hence the SHARED variable is found only in the source procedure.

The following example displays two source procedures, Source1.p and Source2.p that run the consume.p procedure. The value of the SHARED variable sharedVar in consumer.p depends on the source procedure that is being executed.

The following code shows how to create Source1.p:

```plaintext
/* Source1.p : Creating new shared variable */
DEFINE NEW SHARED VARIABLE sharedVar AS CHARACTER.
/* Initialize shared variable */
sharedVar = "This is source1".
/* Statically invoked */
RUN consume.p.
```

The following code shows how to create Source2.p:

```plaintext
/* Source2.p : Creating new shared variable */
DEFINE NEW SHARED VARIABLE sharedVar AS CHARACTER.
/* Initialize shared variable */
sharedVar = "This is source2".
/* Statically invoked */
RUN consume.p.
```

The following code shows how the shared variable is used:

```plaintext
/* consume.p : Defining shared variable */
DEFINE SHARED VARIABLE sharedVar AS CHARACTER.
/* Accessing shared variable value set in source procedure */
MESSAGE sharedVar
VIEW-AS ALERT-BOX.
sharedVar = "02".
```

- **Temp table or DataSet**
You can find references to a temp table or a DataSet defined in the current Procedure (.p) file, Class (.cls) file, AppBuilder (.w) file, or in an Include (.i) file. If the temp table or DataSet is defined in the Include, Class, or Procedure file then the functionality searches for the reference in all the files including the temp tables and DataSets. However, the references to a shared temp table or DataSet are not found as they are available to other procedures only till the procedure that defined it, runs.

The following example displays temp-item temp table including two fields and two indexes. You can find references to the temp table, fields, or index. For example, if you find references to temp-item, PDSOE displays all the references to temp-item including the fields that qualify with the specified temp table.

```abl
DEFINE TEMP-TABLE temp-item
  FIELD cat-page LIKE Item.CatPage
  FIELD inventory LIKE Item.Price LABEL "Inventory Value"
  INDEX cat-page IS PRIMARY cat-page ASCENDING
  INDEX inventory-value inventory DESCENDING.

DEFINE VARIABLE cutoff NO-UNDO LIKE item.price.
DEFINE VARIABLE inv-value NO-UNDO LIKE item.price.
DEFINE VARIABLE report-type AS INTEGER NO-UNDO INITIAL 1.
DEFINE BUTTON ok-butt LABEL "OK" AUTO-GO.
DEFINE BUTTON cancel-butt LABEL "CANCEL" AUTO-ENDKEY.
FORM cutoff LABEL "Inventory Lower Cutoff for each Catalog Page"
  AT ROW 1.25 COLUMN 2 report-type LABEL "Report Sorted ..."
  AT ROW 2.25 COLUMN 2 VIEW-AS RADIO-SET RADIO-BUTTONS "By Catalog Page", 1,
                   "By Inventory Value", 2 SKIP ok-butt cancel-butt WITH FRAME select-frame SIDE-LABELS WIDTH 70 TITLE "Specify Report ..." VIEW-AS DIALOG-BOX.
FOR EACH Item BREAK BY Item.CatPage:
  IF LAST-OF(Item.CatPage) THEN
   DO:
     inv-value = ACCUM SUB-TOTAL BY Item.CatPage
     CREATE temp-item.
     temp-item.cat-page = Item.CatPage.
     inventory = inv-value.
   END.
  END. /* FOR EACH item */
ON CHOOSE OF ok-butt DO:
  HIDE FRAME select-frame.
  IF report-type = 1 THEN
    FOR EACH temp-item USE-INDEX cat-page WITH FRAME rpt1-frame:
      IF inventory >= cutoff THEN
        DISPLAY temp-item.cat-page inventory.
    END.
  ELSE
    FOR EACH temp-item USE-INDEX inventory-value WITH FRAME rpt2-frame:
      IF inventory >= cutoff THEN
        DISPLAY temp-item.cat-page inventory.
    END.
  END.
VIEW FRAME select-frame.
ENABLE ALL WITH FRAME select-frame.
WAIT-FOR CHOOSE OF cancel-butt OR WINDOW-CLOSE OF CURRENT-WINDOW.
```
The following example displays the dsStudent DataSet. When you find references to this DataSet, PDSOE displays all the references to the dsStudent DataSet.

```
p/*Temp-table definition*/
DEFINE TEMP-TABLE ttCourse NO-UNDO
   FIELD CourseID   AS CHARACTER
   FIELD CourseName AS CHARACTER
   FIELD Duration  AS INTEGER
   FIELD ClgID     AS CHARACTER
   FIELD prereqId  AS CHARACTER
INDEX test IS UNIQUE ClgID CourseID.

/*Temp-table definition*/
DEFINE DATASET dsStudent FOR ttCourse
   DATA-RELATION relation FOR ttCourse, ttCourse RELATION-FIELDS
                  (CourseID, prereqId) RECURSIVE.

DATASET dsStudent:HANDLE.
DEF VAR i AS INT.
DEF VAR h AS HANDLE.
DO i = 1 TO DATASET dsStudent:num-buffers:
   h = DATASET dsStudent:get-buffer-handle(i).
   MESSAGE
      h:NAME SKIP
      h:PARENT-RELATION SKIP
      h:NUM-CHILD-RELATIONS SKIP
      VIEW-AS ALERT-BOX.
END.
```

- **Buffer**

  You can find references to a buffer in the current Procedure (.p) file, Class (.cls) file, AppBuilder (.w) file, or in an Include (.i) file. If the buffer is defined in the Include, Class, or Procedure file then the functionality searches for the reference in all the files. However, the reference to a shared buffer is not found.

  The following example displays how cust buffer is used in proc1 internal procedure. When you find references to this buffer, PDSOE displays all the references to the cust buffer.

```
PROCEDURE proc1:
   /*Buffer declaration for Customer table*/
   DEFINE BUFFER cust FOR Customer.
   FOR EACH cust NO-LOCK BY cust.CreditLimit DESCENDING:
      DISPLAY "Highest:" cust.CustNum cust.Name cust.CreditLimit
            WITH 1 DOWN.
      LEAVE.
   END.
   FOR EACH cust NO-LOCK WHERE cust.State = "NH"
      BY cust.CreditLimit DESCENDING:
         DISPLAY cust.CustNum cust.Name cust.CreditLimit.
   END.

END PROCEDURE.
```

- **ABL class level members**

  You can find references to method and variable and class members of a class depending on the access mode and way the members are defined.

- **Class method**

  You can find references to an instance method depending on the access mode. The functionality finds all the references in the same class file (if the class is declared as private), class hierarchy (if the hierarchy is declared as protected), and other publicly accessed classes or procedure files for a particular class method. However, references to the DYNAMIC-INVOKER function, the Progress.Lang.Class.Invoke method, method calls with expression as parameters, or other library files are not found.
The following example displays a sample class including the sayHello instance method and sayHi class method.

METHOD PUBLIC VOID sayHello( ):
   RETURN.
END METHOD.

METHOD PUBLIC STATIC VOID sayHi( ):
   RETURN.
END METHOD.
END CLASS.

The following class uses the Sample class:

CLASS TestClass:
   /*defining variable for Sample class*/
   DEFINE VARIABLE var1 AS Sample NO-UNDO.
   /*-------------------------------*/
   /*Purpose: Test method which calls instance method and class method.
   Notes:*/
   METHODS
   METHOD PUBLIC VOID test( ):
      /*Instance method call*/
      var1:sayHello().
      /*Class level method call*/
      Sample:sayHi().
      RETURN.
   END METHOD.
END CLASS.

• Class and interface types

You can find all occurrences of object-oriented ABL (class and interface) types that are used in a procedure file, include file, class, and interface. You can search for user-defined types and types from libraries. The search for object-oriented ABL types depends on the project’s PROPATH information. Types are resolved based on the PROPATH entries. Types with the same names are resolved by their package name. If the package names are the same, then the types are resolved based on the order of the PROPATH entries.

• Queries

You can find references to all instances of a query definition in a procedure file, include file, class, and interface. If the query definition is in an include file, then you can find references to it in all the files that use it. In case of a procedure, you can find its references in the currently open file. If the query definition is in a class or interface and if the query is declared as a private class, then you can find its references in the currently open file; if it is declared as a protected class, then you can find its references in the hierarchy of that class.

Limitation: In a procedure file, you can declare a query as a shared query, which can be shared by one or more procedures that are directly or indirectly called by the current procedure. The shared query remains available to the other procedures until the procedure that defines it ends. The procedures that are called by the current procedure must define the query with the same name using the DEFINE SHARED TEMP-TABLE statement. When you try to find references of such shared queries, the ABL editor displays only those references that are in the file where the query is defined; the other procedures directly or indirectly accessing the query are not displayed.
The following example illustrates the limitation for finding references to queries. In the example, Source.p and Consumer.p share the query (sharedQry). When the code is executed to find references to sharedQry from Source.p, only references from the current file (Source.p) are displayed; the references from Consumer.p are not displayed.

```abl
/* Source.p : Creating new shared QUERY */
DEFINE NEW SHARED BUFFER sharedBuf FOR Customer.
DEFINE NEW SHARED QUERY sharedQry FOR sharedBuf.
OPEN QUERY sharedQry FOR EACH sharedBuf NO-LOCK.

/* Statically invoked */
RUN shared_query/Consumer.p.

/* Accessing the QUERY shared by Source.p*/
GET FIRST sharedQry.

/*Displaying the Customer details*/
DO WHILE AVAILABLE sharedBuf:
   DISPLAY sharedBuf.CustNum sharedBuf.Name SKIP
   sharedBuf.Phone SKIP sharedBuf.Address
   VIEW-AS EDITOR SIZE 50 BY 2 SCROLLBAR-VERICAL
   WITH FRAME ord-info CENTERED SIDE-LABELS TITLE "Order Information".
   PAUSE.
   GET NEXT sharedQry.
END.

/* DO WHILE AVAILABLE records */
```

- **Class and interface types**
  You can find all occurrences of object-oriented ABL (class and interface) types that are used in a procedure file, include file, class, and interface. You can search for user-defined types and types from libraries. The search for object-oriented ABL types depends on the project's PROPATH information. Types are resolved based on the PROPATH entries. Types with same names are resolved by their package name. If the package names are the same, then the types are resolved based on the order of the PROPATH entries.

- **Properties**
  You can find references to the ABL properties in a project or a workspace. You can search for the references to the properties that are defined in the current file or in the hierarchy of the current file. You can also find references to properties accessed statically or through class instances.

- **ABL events**
  You can find references to ABL events in a project or a workspace. You can search for the references to events that are defined in the current file or in the hierarchy of the current file. You can also find references to the events accessed statically or through class instances.

- **Data-sources**
  You can find references to ABL data-sources in a project or a workspace depending on where you defined the data-source. If the data-source is defined in an Include (.i) file, all the references to the files including that Include file are found. If it is defined in a Procedure (.p) or an AppBuilder (.w) file, only the reference to the current file is found. If it is defined in a Class (.cls) file, the references to the current class file (if the data-source is declared as private) or class hierarchy (if the data-source is declared as protected) are found.
**ABLDoc documentation generation**

You can generate API documentation (ABLDoc) in HTML format (default) from ABL source code for your reference to your codes outside OpenEdge. You can generate the ABLDoc documentation for the following source codes:

- Class (.cls) files
- Procedure (.p) files
- Include (.i) files

You can generate the ABLDoc documentation using the Generate ABLDoc wizard or by writing ANT tasks for it.

**See also**
- Generating ABLDoc documentation on page 425
- Generate ABLDoc wizard on page 463
- Navigating through the ABLDoc Navigation page on page 429
- Adding comments on page 430
- Customizing ABLDoc output on page 431

**The OpenEdge Editor perspective**

The default OpenEdge Editor perspective contains several views. Some are standard Workbench views. Some are specific to Progress Developer Studio for OpenEdge. As with all perspectives, you can add or delete views from the OpenEdge Editor perspective. If you want to reset to the default perspective configuration, choose Window > Reset Perspective.

By default, the OpenEdge Editor perspective includes the following views:

- **ABL Editor** - A code editor customized for working with ABL. You can have multiple ABL source files open simultaneously in separate editor buffers.
- **Project Explorer** - A hierarchical view of the projects and resources in your Eclipse workspace.
- **Outline** - A list of the structural elements of the file that has focus in the ABL Editor.
- **DB Structure** - An OpenEdge view that displays the schema of connected databases. You can drag and drop schema elements (tables, fields, and others) from this view into a file open in the ABL Editor.
- **Console** - A Workbench view that displays the text output from commands (run-time startup, for example).
- **Problems** - A log of the errors, warnings, and other information associated with a file that has been opened in the ABL editor.
- **Tasks** - A to-do list.

**See also**
- ABL Resources in Progress Developer Studio for OpenEdge on page 364
- ABL Editor Features on page 368
- Using the Outline view on page 399
- Working with databases on page 395
- Introducing the ABL Editor on page 363
Tasks

Setting Editor options

Setting Eclipse editor preferences

The OpenEdge Editor is the standard Eclipse text editor customized for working with ABL. The OpenEdge Editor inherits behavior from the preferences you set for all text editors.

To set these general editor preferences, choose Windows > Preferences > General > Editors. On these preference pages, you can set file associations, tab spacing, line numbers, the Eclipse “Quick Diff” functionality, and other features that affect all text editors in the Workbench. For information on the Eclipse text editor preferences, see the Editors topic in the Workbench User Guide.

Eclipse lets you compare text files by selecting the files in the Project Explorer view, right-clicking, and selecting Compare With from the context menu.

You can choose the font used for the display of the files being compared.

To choose the font:

1. Select Window > Preferences > General > Appearance > Colors and Fonts.

2. In the list in the right pane, expand the Text Compare node and select ABL Compare Editor font.

3. Do one of the following:
   • If you want to use the default system font styles, click Use Style, Font.
   • If you want to select your own font styles, click Edit. The Font window appears. Go to Step 4 on page 379.
   • If you want to reset the font settings to default, click Reset.

4. Select the font styles and click OK.

See also

Setting ABL Editor preferences on page 379
Associating custom file extensions with the ABL Editor on page 380
ABL Editor preferences on page 433

Setting ABL Editor preferences

Use preference settings to modify the behavior of the OpenEdge ABL Editor.

To modify the behavior of the ABL Editor:

1. Select Window > Preferences. The Preferences window appears.

2. Select the Progress OpenEdge > Editor node. The Editor options page appears with settings for indentation style, keyword case, and automatic expansion of keywords and schema references. See also Editor options.
3. Expand the Editor node to select from the following pages:

- **Annotations** - A tool for creating and managing annotation templates
- **Assistance** - Options for syntax help and code completion
- **Build** - Options for how the Editor handles files with certain extensions
- **Colors** - Options for color-coding of syntax elements
- **Templates (Macros)** - A tool for creating and managing macro definitions for recalling stored strings

4. Once you have made changes to the Editor preference settings, click **Apply** to save your changes.

   **Note:** Click **Restore Defaults** to reset Editor preferences to their default values.

---

**See also**

[Setting Eclipse Editor preferences on page 379](#)
[Associating custom file extensions with the ABL Editor on page 380](#)
[ABL Editor preferences on page 433](#)

---

**Associating custom file extensions with the ABL Editor**

If you use custom file extensions for your OpenEdge files, you need to associate those extensions with the ABL Editor. To do so, you set both general Eclipse editor preferences and OpenEdge-specific Editor preferences:

1. Choose **Window > Preferences**. The **Preferences** window appears.
2. Choose **General > Editors > File Associations**. The **File Associations** page appears.
3. Click **Add for the File types list**. The **New File Type** dialog appears.
4. Type your custom file extension and click **OK**.
5. Select your file extension in the File types list.
6. Click **Add for the Associated editors list**. The **Editor Selection** dialog appears.
7. Select one of the OpenEdge ABL Editor options and click **OK**.

   **Note:** The only difference among these options is the icon used to represent the files that it opens. All the choices are functionally the same.

8. Select **Progress OpenEdge > Editor > Build** in the **Preferences** window. The **Build** page appears.
9. Add your custom file extension to the appropriate list and click **Apply** to save your changes.

---

**See also**

[Setting ABL Editor preferences on page 379](#)
[Setting Eclipse Editor preferences on page 379](#)
[Editor Build options on page 438](#)
Writing ABL code

Demos
Using the Quick Diff Editing Feature
Using the History View

Note: The OpenEdge Release 10.2B demos have not been updated for OpenEdge Release 11.0. Since 10.2B, OpenEdge Architect has been renamed as Progress Developer Studio for OpenEdge. In addition, OpenEdge Release 11.0 includes some feature enhancements and other changes. Despite the differences between 10.2B and 11.0, the demos are still a useful introduction to Progress Developer Studio for OpenEdge.

Creating ABL source files

Progress Developer Studio for OpenEdge provides wizards for creating commonly used ABL file types:

- Class
- Include file
- Interface
- Procedure
- Structured include file (contains AppBuilder markup)
- Structured procedure (contains AppBuilder markup)
- ABL Form (Windows only)
- ABL Dialog (Windows only)
- ABL MDI Form (Windows only)

These wizards create files with placeholders and default values and comments for commonly used code sections, simplifying the process of setting up a variety of source files. If you prefer, you can create a new blank file and enter all of the content yourself. As long as you use a file extension that is associated with the ABL Editor, you can use the ABL Editor functions when working with it.

To create a new source file:

1. Open the OpenEdge project in which you plan to use the file.
2. Choose File > New. If the desired file type does not appear in the menu, choose Other > Progress OpenEdge > Editor.
3. Select the desired file type and click Next to launch the corresponding wizard.
4. Fill in the appropriate information for your new file.
5. Click Finish. The file opens in an ABL Editor window.

See also
ABL Resources in Progress Developer Studio for OpenEdge on page 364
Writing procedural code on page 383
Writing object-oriented code on page 385
Using comments

You can quickly "comment out" and uncomment blocks of your code. Commented blocks are ignored by the compiler and not treated as part of the program syntax.

To use this ABL Editor feature:

1. Select the lines that you want to comment or uncomment.
2. Right-click and select **Source > Toggle Comment or type CTRL+*/**.

   If the selected block is not commented, the comment-delimiter characters (/* and */) are inserted at the beginning and end of the block. If the selected block is commented, the delimiter characters are removed from the beginning and end of the block. In both cases, the ABL Editor changes the text color to match the settings in the Editor Color preferences.

See also

**Editor Colors options** on page 439

Using templates for easier text entry

Templates are custom character sequences that the ABL Editor automatically expands into ABL statements. Using editor templates, you can facilitate programming tasks by reducing the amount of typing required. For example, you can type "DVCH" to enter a template definition for a character variable into a file: "DVCH" expands to DEFINE VARIABLE AS CHARACTER NO-UNDO and positions the cursor after "VARIABLE" to allow you to enter the variable name. The ABL Editor includes predefined macro definitions, referred to as templates in the Eclipse environment, that automatically expand to more complex strings. You can add, modify, and delete macros through the **Templates** preferences page.

**Note:** Templates were formerly called macros in Progress Developer Studio for OpenEdge, and are referred to as aliases in some OpenEdge applications, such as AppBuilder. Macros created in versions earlier than OpenEdge 10.1C are not compatible with the current version and must be migrated to templates before you can use them. If you have such macros in your workspace, select **OpenEdge > Migration > Migrate 10.1A or 10.1B Macros to Templates** to launch the migration tool.

1. To use a template (macro) while working in an ABL source file, type the template name and press SPACE to expand it to the defined full string.

2. To manage the set of available templates, go to the **Templates (Macros)** page in Editor preferences:
   a) Choose **Window > Preferences**. The **Preferences** window appears.
   b) Choose **Progress OpenEdge > Editor > Templates (Macros)**. The **Templates** page appears. The page displays the list of existing templates.
   c) To modify a template, click **Edit** and make the desired changes. Click **Apply** to save your changes.

3. At this page, you can also click **New** to create templates, click **Import** to add templates from other workspaces, or click **Export** to make templates available for use in other workspaces.
Note: Macros created in versions earlier than OpenEdge 10.1C are not compatible with the current version and must be migrated to templates before you can use them. If you have such macros in your workspace, select OpenEdge > Migration > Migrate 10.1A or 10.1B Macros to Templates to launch the migration tool.

See also
Using assistance features on page 389
Editor Templates (Macros) options on page 440

Comparing files
To compare ABL files in Progress Developer Studio for OpenEdge:

1. Select two files in the Project Explorer view.
2. From the context menu, select Compare With > Each Other. The ABL files appear in the Eclipse Compare Editor. The files are color-coded as in the ABL Editor.

Note: You can compare three files if one file is the common ancestor of the other two.

Removing AppBuilder markup
Files created in AppBuilder contain code that enables it to parse the file. If you remove these statements, you cannot edit the file in AppBuilder. However, if you no longer intend to maintain a file with AppBuilder, these statements serve no further purpose.

You can remove these statements by choosing Source > Strip AppBuilder markup from the ABL Editor context (right-click) menu.

See also
AppBuilder code on page 366

Writing procedural code

Adding functions
The OpenEdge Editor provides a wizard that helps you add a new function to an ABL source file. To use this wizard:

2. If the current file is an ABL procedure file, then go to step 3.
   
   Otherwise, if the current file is an AppBuilder procedure (adm1 or adm2) file, do one of the following:
   
   • Select the Function option to create a function.
   • Select the Override option to have the Name field display a list of the user-defined functions that you can override.

Note: The Override option is disabled when there are no functions to override.
3. In the **Name** field, specify a function name.

   **Note:** If you have selected the **Override** option, the **Name** field displays a list of user-defined functions that you can override. Select a function that you want to override.

4. Select a return type, and select whether the function is private. Also choose whether to include a **CATCH** block and/or a **FINALLY** block for structured error-handling.

5. Choose an insertion position. The Insertion position field gives you the following options:

   - **Alphabetical order** - Insert a code prototype based on its name. Case is ignored.

      **Note:** This feature does not sort the procedures, functions, methods, or properties that already exist in the file. If they are already ordered alphabetically by name, insertion of new code will be done at the correct position in the list. Otherwise, insertion will be the first valid alphabetic position found during a top-to-bottom lookup.

   - **Cursor position** - Insert code prototype at the current position of the cursor in the file.

   - **First/Last** - Insert the code prototype as the first or last item

   - **After name** - Insert the code prototype after the named procedure.

6. Click **OK**.

   The OpenEdge Editor inserts a prototype for the function. You can fill in and modify the prototype as appropriate.

**See also**

- Support for structured error handling on page 370
- Add Procedure wizard on page 455
- Add Procedure wizard on page 455

**Adding internal procedures**

The OpenEdge Editor provides a wizard that helps you add internal procedures to a procedure file. To use this wizard:

1. Select **Source > Add Procedure**. The **Add Procedure** dialog appears.

2. If the current file is an ABL procedure file, then go to step 3.

   Otherwise, if the current file is an AppBuilder procedure (adm1 or adm2) file, do one of the following:

   - Select the **Procedure** option to create a procedure.

   - Select the **Override** option to have the **Procedure name** field display a list of the procedures for which you can create overrides.

      **Note:** This list is the internal procedures of all or Super Procedures defined for the procedure file. This option is disabled when there are no procedures to override.

3. In the **Procedure name** field, specify a procedure name. Also choose whether to include a **CATCH** block and/or a **FINALLY** block for structured error handling.

4. Choose an insertion position.
• **Alphabetical order** - Insert a code prototype based on its name. Case is ignored.
• **Cursor position** - Insert code prototype at the current position of the cursor in the file.
• **First/Last** - Insert the code prototype as the first or last item.
• **After name** - Insert the code prototype after the named procedure.

5. Click **OK**.

The OpenEdge Editor inserts the internal procedure.

**See also**
- Support for structured error handling on page 370
- Adding functions on page 383
- Add Procedure wizard on page 455

## Writing object-oriented code

<table>
<thead>
<tr>
<th>Demo</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Navigating Classes in Progress Developer Studio for OpenEdge, Part 1: Using the ABL Editor</strong></td>
</tr>
</tbody>
</table>

**Note:** The OpenEdge Release 10.2B demos have not been updated for OpenEdge Release 11.0. Since 10.2B, OpenEdge Architect has been renamed as Progress Developer Studio for OpenEdge. In addition, OpenEdge Release 11.0 includes some feature enhancements and other changes. Despite the differences between 10.2B and 11.0, the demos are still a useful introduction to Progress Developer Studio for OpenEdge.

### Adding constructors

When you create a new class, you can add a default constructor by selecting the **Generate default constructor** toggle in the **New ABL Class** wizard.

You can add constructors to an existing class as follows:

1. Choose **Source > Add Constructor** to launch the **Add Constructor** wizard.
2. Select the appropriate access modifier (**Public**, **Protected**, or **Private**).
3. Choose whether to include a CATCH block and/or a FINALLY block for structured error handling.
4. Select the insertion position for the constructor and click **Generate**. The constructor appears at the specified position in your class file.

**Note:** You can also add a static constructor to an existing class by choosing **Source > Add Static Constructor**.

**See also**
- Object-oriented code on page 365
- Support for structured error handling on page 370
- Adding constructors from a super class on page 452
- Adding a destructor on page 386
Adding constructors from a super class
The object-oriented extensions of the ABL enable you to overload constructors from a super class. To do so:

1. Choose Source > Add Constructors from super class to launch the Add Constructors from Super Class wizard.
2. Select the constructors to overload from the list of available constructors in the super class.
3. Click Generate. The constructors are added to the code at the specified insertion position.

Note: This option is disabled if the super class includes no constructors with parameters.

Adding a destructor
When you create a new class, you can add a destructor by selecting the Generate destructor toggle in the New ABL Class wizard. You can add a destructor to an existing file by choosing Source > Add Destructor. The editor adds the destructor at the current cursor position.

Note: A class can have only a single destructor.

Adding methods
You can add methods to a class as follows:

1. Choose Source > Add Method to launch the Add Method wizard.
2. Fill in the appropriate information for the new method. Help text appears in the title pane to describe the purpose of the field that has focus.
3. Click Generate. Progress Developer Studio for OpenEdge inserts the method at the place specified in the Insertion position field. You have the following options:
   - **Alphabetical order** - Insert the code prototype at a position determined by its name. Case is ignored.
Note: This feature does not sort the procedures, functions, methods, or properties that already exist in the file. If they are already ordered alphabetically by name, the code is inserted at the correct position in the list. Otherwise, the code is inserted at the first valid alphabetic position found during a top-to-bottom lookup.

- **Cursor position** - Insert the code prototype at the current position of the cursor in the file.
- **First/Last** - Insert the code prototype immediately before/after the first/last item currently in the class.
- **After name** - Insert the code prototype after the named item.

See also
Object-oriented code on page 365
Overriding members on page 388
Coding methods implemented via interface or abstraction on page 389
Add Method wizard on page 454

Adding events
You can add events to a class as follows:

1. Choose **Source > Add Event** to launch the **Add Event** wizard.
2. Fill in the appropriate information for the new event. Help text appears in the title pane to describe the purpose of the field that has focus.
3. Click **Generate**. Progress Developer Studio for OpenEdge inserts the event at the place specified in the **Insertion position** field. You have the following options:
   - **Alphabetical order** - Insert the code prototype at a position determined by its name. Case is ignored.

   Note: This feature does not sort the procedures, functions, methods, or properties that already exist in the file. If they are already ordered alphabetically by name, the code is inserted at the correct position in the list. Otherwise, the code is inserted at the first valid alphabetic position found during a top-to-bottom lookup.

   - **Cursor position** - Insert the code prototype at the current position of the cursor in the file.
   - **First/Last** - Insert the code prototype immediately before/after the first/last item currently in the class.
   - **After name** - Insert the code prototype after the named item.

See also
Object-oriented code on page 365
Add Event wizard on page 452

Adding properties
You can add properties to a class as follows:
1. Select **Source > Add Property** to launch the **Add Property** wizard.

2. Fill in the appropriate information for the new property. Help text appears in the title pane to describe the purpose of the field that has focus.

3. Click **Generate**. The editor inserts the property where you specify. The **Insertion position** field gives you the following options:

   - **Alphabetical order** - Insert the code prototype at a position determined by its name. Case is ignored.

     **Note:** This feature does not sort the procedures, functions, methods, or properties that already exist in the file. If they are already ordered alphabetically by name, the code is inserted at the correct position in the list. Otherwise, the code is inserted at the first valid alphabetic position found during a top-to-bottom lookup.

   - **Cursor position** - Insert the code prototype at the current position of the cursor in the file.
   - **First/Last** - Insert the code prototype immediately before/after the first/last item currently in the class.
   - **After name** - Insert the code prototype after the named item.

**See also**

Object-oriented code on page 365
Add Property wizard on page 456

**Overriding members**

The OpenEdge Editor includes an option to ease the task of finding the correct methods, events, and properties to override in your class hierarchy.

1. Select **Source > Override/Implement Members** to launch the **Override/Implement Members** wizard.

2. Select the members to override (or implement) from the list of members available in the class hierarchy.

3. Select the desired insertion position.

4. Click **Generate**. The overrides are added at the specified position in the source file.

**Note:** A green-triangle decorator ✴️ appears on the marker bar in the ABL Editor next to any member that is an override or overload.

**See also**

Object-oriented code on page 365
Adding methods on page 386
Adding events on page 387
Adding properties on page 387
Override/Implement Members wizard on page 463
Coding members implemented via interface or abstraction

If you specify one or more interfaces when creating a new ABL class, the class must include implementations for all methods, events, and properties declared in those interfaces; otherwise, the class cannot be compiled. Similarly, if the class inherits one or more abstract members from an abstract super class, those members must be implemented (unless the inheriting class is also abstract).

To help you satisfy these requirements, the ABL Editor automatically generates stubs for such members when you create the class. For existing class files that do not include the required members, you can use the Override/Implement Members wizard to add them.

You choose whether method stubs added to the file are to include error code, or default return values.

Method stubs automatically generated in new class files

When you use the New ABL Class wizard (or any of the Visual Designer form-creation wizards) to create a class file that implements an interface or inherits from an abstract super class, select one of the following options for the method code:

- Throw a Not Implemented exception
- Return default values

Note that the wizard automatically creates stubs for all of the applicable members, with the method stubs coded according to the option that you select.

Method stubs generated on demand in existing class files

While editing a class file, you can add stubs for any missing members implemented via interface or inherited from an abstract super class. To do so:

1. Select **Source > Override/Implement Members**.

   **Note:** The tree in the Override/Implement Members wizard lists all methods, events, and properties declared in the applicable interfaces and/or in the abstract super class but missing from the source file. The members are selected by default for generating stubs. You can deselect any methods for which you do not want to add stubs at this time.

2. Select either Generate exceptions for required members or Generate default values for required members.

3. Click Generate to add the selected member stubs to the class file.

See also

Object-oriented code on page 365
Override/Implement Members wizard on page 463
New ABL Class wizard on page 458

Using assistance features

Using context assistance

Context assistance is non-interactive information about the ABL code base. It does not offer typing shortcuts as code-completion assistance does.
Showing syntax help as you type

To show syntax help, type an ABL keyword followed by a space. A list of phrases valid for the context appears in a pop-up window.

This feature is enabled by default. To turn it off or on:

1. Choose Window > Preferences to open the Preferences window.
2. Choose Progress OpenEdge > Editor > Assistance to display the Assistance page.
3. Uncheck or check the Show syntax help as I type option.

Note: You can quickly display full entries from OpenEdge Development: ABL Reference in the Eclipse help viewer by Highlight a keyword or phrase in the ABL Editor and Press SHIFT+F2 or right-click and choose Keyword Help from the context menu.

See also

Code assistance on page 369
Using bracket-matching on page 390
Using hover help on page 391
Editor Assistance options on page 436

Using bracket-matching

Bracket-matching provides visual cues illustrating the location of matched pair elements. With bracket-matching enabled, positioning the cursor on one paired element highlights the matching element in the code sequence.

Bracket-matching works with:

• Double and single quotation marks
• Brackets and parentheses ( "", "[", "(")
• Code blocks
• DO, FOR, REPEAT
• FUNCTION, PROCEDURE, METHOD

The code example below illustrates highlighting of the matching element (in this case, a closing parenthesis):

```abl
IF DataObject <> "" THEN DO;
  setAppService('');
  startDataObject(DataObject).
  DataSourceId = dynamic-function("getDataSource").
END.
ELSE DO;
```

Note: Bracket-matching depends on the parsing of the code being up to date. In some circumstances, notably in the case of very large files or projects connected to remote databases, updating the analysis of the code can take several seconds or more. If the code has not been fully analyzed within a user-specified interval (by default 300 milliseconds), the bracket-matching feature is temporarily disabled automatically. The feature is automatically re-enabled as soon as parsing is up to date again.
With the cursor in or immediately to the right of a paired element, press `CTRL+SHIFT+P` to go to the matching element to jump to a matching element. For example, using this key sequence with the cursor in a DO statement repositions the cursor to the right of the corresponding END statement.

Disabling bracket-matching does not disable jumping to the matching element with `CTRL+SHIFT+P`.

This feature is enabled by default. To turn it off or on:

1. Choose **Window > Preferences** to open the **Preferences** window.
2. Choose **Progress OpenEdge > Editor > Assistance** to display the **Assistance** page.
3. Uncheck or check the **Highlight matching elements in code** option.

**See also**
- [Code assistance on page 369](#)
- [Showing syntax help as you type on page 390](#)
- [Using hover help on page 391](#)
- [Using code-completion assistance on page 393](#)
- [Editor Assistance options on page 436](#)

**Using hover help**

Hover help displays information about database fields, system-handles, and ABL statements when you position the cursor over a particular element in the code.

You can disable this feature in OpenEdge Editor Assistance preferences (**Windows > Preferences > Progress OpenEdge > Editor > Assistance**).

**Note:** This feature relies on Progress Developer Studio for OpenEdge's grammar parsing functionality. The text hover appears only for valid syntax. In some cases, a particular use of a keyword might not be indexed separately. Try hovering over another keyword in the statement.

The illustration below provides an example of hover help for a **database** field:

```
SPORTS2000.Customer.Address
```

The customer table contains customer information including balance and address.

System handles also support hover help, as illustrated below:

```
IF ERROR-STATUS:ERROR THEN DO:
  ERRCR-STATUS - system-handle
  A handle to error information on the last statement executed with the NO-ERROR option.
  ERRCR-STATUS [:attribute][:method]
```

For ABL statements, keywords are treated as multiple parts that together form a statement. For example, **DEFINE** by itself does not provide hover help; however, when coupled with a type, such as **DEFINE VARIABLE**, hover help provides additional information:
Many optional parameters in ABL statements possess different meanings depending on the statement to which they belong. The hover help for these options and parameters varies based on their use in a particular statement. The FOR statement is one such example. This statement can act as a standalone element, or it can serve as an optional parameter for other ABL statements, including COPY-LOB, DEFINE, BUFFER, DEFINE DATA-SOURCE, DEFINE DATASET, DEFINE PARAMETER, DEFINE QUERY, DO, FUNCTION, OPEN QUERY, or REPEAT.

The illustration below shows text hover on an AS element in the context of a DEFINE VARIABLE statement:

```
DEFINE VARIABLE CurrentRows AS CHARACTER NO-UNDO.
```

Text hover is also applied on attributes and methods:

```
THIS-PROCEDURE:ADD-SUPER-PROCEDURE (li)
```

The illustration below provides an example of hover pop-up for the DYNAMIC-PROPERTY function:

```
DYNAMIC-PROPERTY (dynobj, dynProp)
```

The illustration below provides an example of hover pop-up for a preprocessor reference:
When you hover over a preprocessor reference, the value of the preprocessor is displayed in a pop-up dialog. For example, in the above illustration, hovering over the preprocessor {&TABLE} displays the current value of preprocessor as Customer in a pop-up dialog. If the preprocessor evaluation is not completed at the time of hovering, no value is displayed in the pop-up dialog.

See also
Code assistance on page 369
Showing syntax help as you type on page 390
Using bracket-matching on page 390
Editor Assistance options on page 436

Using code-completion assistance
Code-completion assistance can help you save time and reduce typing errors by having the ABL Editor insert syntax elements for you.

Invoking syntax-completion assistance
Syntax-completion assistance proposes syntax to complete the code that you are typing in the ABL Editor. When you press CTRL+SPACE, proposals appear in the left pane of a pop-up window, with reference information about the selected item in the right pane, as in this example:

```
DEFINE
  ABSTRACT - OPTION
  BROWSE - OPTION
  BUFFER - OPTION
  BUTTON - OPTION
  DATASOURCE - OPTION
  DATASET - OPTION
  EVENT
  FRAME - OPTION
  IMAGE - OPTION
  INPUT - OPTION
```

To insert an element at the current cursor position, select it and double-click or press ENTER.

Press CTRL+SPACE to get proposals in any of these contexts:

- ABL keywords
- Object variable and property names that are reserved keywords
- Object references for the CREATE statement
- Procedures names and paths for the RUN statement (based on PROPATH)
- Widget variables and handles
- Parameters of procedures and methods
- Table names for connected databases
• Fields for temp-tables, buffers, and datasets
• Preprocessor names
• Subscripted array references

By default, syntax-completion assistance filters proposals based on context (for example, showing only relevant keywords). Press **CTRL+SPACE** while the pop-up window is open to toggle between context-filtered proposals and all proposals.

In addition to invoking completion assistance at any time by pressing **CTRL+SPACE**, you can have the ABL Editor propose completion options automatically when you type a period in a schema reference, or when you type a colon in a handle or object reference. To enable this feature, select the **Automatically propose completion on '.' and ':'** option on the **Editor Assistance Preferences** page.

If you want to see only certain types of syntax-completion proposals, you can suppress the others. Under **Exclude proposals** on the **Editor Assistance Preferences** page, check the categories that you do not want to see.

**See also**

- Code assistance on page 369
- Balancing code-completion convenience and performance on page 395
- Using context assistance on page 389
- Editor Assistance options on page 436

**Using auto-bracketing**

With auto-bracketing enabled, when you entering the first of two paired elements, the matching element is automatically inserted to the right of the cursor. Auto-bracketing works with:

• Double and single quotation marks
• Brackets and parentheses ( "{", "[", "(")

Note the following:

• When you type a single or a double quote mark, the ending quote mark is automatically added. If you start typing in between quote marks and add another ending quote mark, the additional quote mark is not added.

• You can delete an ending character that is automatically added, for example an ending quote mark, by pressing Backspace.

• If you want to insert a quote mark inside existing quote marks, type ~ and add another quote mark.

To enable the insertion of matching elements:

1. Choose **Window > Preferences** to open the **Preferences** window.
2. Choose **Progress OpenEdge > Editor > Assistance** to display the **Assistance** page.
3. Select the **Auto-bracketing** option.

**See also**

- Code assistance on page 369
- Balancing code-completion convenience and performance on page 395
- Using context assistance on page 389
- Editor Assistance options on page 436
Balancing code-completion convenience and performance

To propose a complete and accurate set of completion options, Progress Developer Studio for OpenEdge must completely analyze all code and schema information in the current ABL Editor buffer. This analysis is an ongoing process, and in many cases the task is completed so rapidly that there is scarcely any noticeable delay between a request for completion assistance and the response to the request. Under certain circumstances, however, the analysis may take several seconds or more, long enough to make waiting for the list of valid options an annoyance. Editing files that are thousands of lines long, and writing code that refers to a database located on a remote system, are examples of situations that can cause an unacceptably slow system response.

Progress Developer Studio for OpenEdge lets you decide how long you are willing to wait for complete, up-to-date code-completion assistance. In Editor Assistance preferences, the field labeled Milliseconds allowed for re-parsing specifies the maximum amount of time (by default, 300 milliseconds) the ABL Editor will spend updating its analysis before building the list of completion proposals based on the last completed analysis. If this interval elapses before the updated analysis is finished, the completion options that you receive may not be entirely accurate.

The default time allowance of 300 milliseconds should be fine for working on smaller files with local databases, but if your code base requires a relatively long time to be fully parsed, you might want to increase the setting. Note, however, that you cannot enter text while waiting for completion proposals, so a high setting for this value can potentially result in noticeable interruptions to your work.

By default, when the ABL Editor proposes completion options for schema references, it includes field descriptions in the right pane of the pop-up window. In the case of a remote databases, retrieving this information is a time-consuming process that can degrade performance. Therefore, you can choose to disable the inclusion of field descriptions. To do so, deselect the Show description when proposing schema elements option in Editor Assistance preferences.

See also
- Code assistance on page 369
- Using context assistance on page 389
- Editor Assistance options on page 436

Working with databases

Connecting databases to the project

Most ABL applications work in conjunction with one or more databases, and it is important to define connections between each of your projects and the databases that it will work with. By doing so, you enable the ABL Editor to provide complete syntax checking, code assistance with schema references, and drag-and-drop entry of schema references from the DB Structure view.

To connect a project to a database, follow the procedures detailed in the Progress Developer Studio for OpenEdge's help.

See also
- Using the DB Structure view on page 395
- Entering schema references on page 396

Using the DB Structure view

The DB Structure view, included in the default OpenEdge Editor perspective, shows a tree-view outline of the schema of each database connected via a SQL connection profile to the current project. The view includes a separate tab for each database. The nodes are expandable and collapsible.
To enter a schema reference in a source file, select a table or column in the tree view and drag it to the ABL Editor buffer containing the file.

Note: To show the database schema in the DB Structure view, you must use a SQL connection. ABL database connections do not provide this functionality. You can define both an ABL connection and a SQL connection in the connection profile for the database.

If the DB Structure view is not visible in your current perspective, open it by selecting Window > Show View > Other > Progress OpenEdge Editor > DB Structure.

See also
- Connecting databases to the project on page 395
- Entering schema references on page 396

Entering schema references
Several ABL Editor and Progress Developer Studio for OpenEdge features are available to help you enter database schema references in your code:

- **DB Structure view** - Provides a tree-view outline of the schema, with drag-and-drop functionality to enter table and column references in the source code. This feature requires a SQL connection to the database.

- **Expansion of schema element names** - Optionally completes the abbreviated names of tables and columns when you press the space bar. You can turn this feature on or off at the Editor preferences page.

- **Prefixing of database names** - Optionally adds the database name when you type a table name. You can turn this feature on or off at the Editor preferences page.

- **Schema descriptions in code-completion assistance** - Optionally displays the description for each table or column proposed as a completion phrase. Enabling schema descriptions can have a noticeable performance impact in certain situations, such as the use of a large remote database. You can turn this feature on or off at the Editor Assistance preferences page.

- **Hover help** - Optionally shows syntax help when you pause the cursor over a schema reference. You can turn this feature on or off at the Editor Assistance preferences page.

See also
- Connecting databases to the project on page 395
- Using the DB Structure view on page 395
- Setting ABL Editor preferences on page 379
- Editor options on page 433
- Editor Assistance options on page 436

Formatting and displaying code

Using code folding
Code folding temporarily hides sections of your code, such as include files, internal procedures, functions, and methods.

If you can fold a section of code, a symbol (minus sign in a blue circle) appears in the Marker Bar beside the first line of the section. Click this symbol to collapse the display of the section. When collapsed, you can display the hidden code by hovering the cursor over the icon:
After you fold the code, the first line of the section remains and the symbol changes to ✺ (plus sign in a blue circle). Click the symbol to expand the section. See Formatting and display features for more details.

Applying keyword casing

To apply your keyword casing preference (upper or lower, as specified at the Editor preferences page) to the current file:

1. Optionally, select a block of code to which you want to apply keyword casing. If no text is selected, the entire file will be formatted.
2. Press CTRL-SHIFT-F or select Source > Correct Case.

Note: At the Editor preferences page, you can turn on or off options to have the ABL Editor automatically apply keyword casing as you type and when you save the file.

See also

Formatting and display features on page 368
Editor options on page 433

Expanding abbreviated keywords

To expand abbreviated keywords in the current file to their fully spelled equivalents (for example, to change DEF VAR to DEFINE VARIABLE):

1. Optionally, select a block of code in which you want to expand keywords. If no text is selected, the entire file will be formatted.
2. Press CTRL-SHIFT-1 or select Source > Expand Keywords.

Note: At the Editor preferences page, you can turn on or off the option to have the ABL Editor automatically expand keywords as you type.

See also

Formatting and display features on page 368
Editor options on page 433
Indenting and aligning code

<table>
<thead>
<tr>
<th>Demo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indentation and tabular formatting</td>
</tr>
</tbody>
</table>

**Note:** The OpenEdge Release 10.2B demos have not been updated for OpenEdge Release 11.0. Since 10.2B, OpenEdge Architect has been renamed as Progress Developer Studio for OpenEdge. In addition, OpenEdge Release 11.0 includes some feature enhancements and other changes. Despite the differences between 10.2B and 11.0, the demos are still a useful introduction to Progress Developer Studio for OpenEdge.

To consistently indent lines in the current file for improved readability:

1. Optionally, select a block of code to be indented. If no text is selected, the entire file will be formatted.
2. Press **CTRL-I** or select **Source > Correct Indentation**.

**Note:** At the **Editor preferences** page, you can turn on or off options to have the ABL Editor automatically indent and align lines as you type and when you paste text, and to enable tabular formatting. The **Correct Indentation** command also applies tabular formatting, if enabled.

See also
- Formatting and display features on page 368
- Editor options on page 433

Selecting display color for code element

You can use the **Colors** page to select the change colors for different code elements in the ABL Editor.

To change color for a code element:

1. From the main menu, select **Window > Preferences**. The **Preferences** window appears.
2. Select **Progress OpenEdge > Editor > Colors**. The **Colors** page appears.
3. Select a display color for a code element by clicking the color button. The **Color** dialog appears.
4. Do one of the following:
   a) Select from the list of available basic or custom colors.
   b) Click **Define Custom Colors** to define a custom color from the color palette.
5. Click **OK** to save your changes.

See also
- Formatting and displaying code on page 368
- Editor Colors options on page 439
Navigating

Using the Outline view

The Outline view shows the structure of the code in the ABL Editor buffer that currently has focus, and provides an easy way to navigate to specific places in the code. The elements inside the inactive preprocessor regions appear in gray in the Outline view. This is similar to include nodes but without the include decorator.

If the Outline view is not visible in your current perspective, open it by selecting Window > Show View > Outline.

The Outline view supports the following navigation techniques:

• Click an element labeled in black type of position the cursor at the declaration of that element. Black elements are declared in the current file.

  **Note:** To move to a declaration in the current file, you must enable the **Link with Editor** option. Turn this option on or off by selecting it in the drop-down menu that appears when you click **View Menu** on the Outline view toolbar.

• Double-click a black include file name to open that include file.

• Double-click an element labeled in gray type to open the include file and position the cursor at the declaration of that element. Gray elements are declared in include files.

The Outline view toolbar contains the following buttons:

| ![Use Expand Include File Contents](icon) | Use **Expand Include File Contents** to display elements in include files. By default, the Outline view does not show elements declared in include files.  
  **Note:** When Preprocessing option is enabled, include elements are expanded by default and **Expand Include File Contents** appears as disabled. |
| ![Use Collapse All](icon) | Use **Collapse All** to collapse all expanded tree nodes in the Outline view. |
| ![Use Filters](icon) | Use **Filters** to hide the elements of the inactive preprocessed regions from the Outline view. See also Viewing preprocessors in the Outline view.  
  Turn this option on or off by selecting (or clearing) it in the drop-down menu that appears when you click **View Menu** on the Outline view toolbar. |
| ![Enable Link with Editor](icon) | Enable **Link with Editor** to move to a declaration in the current file. Turn this option on or off by selecting (or clearing) it in the drop-down menu that appears when you click **View Menu** on the Outline view toolbar. |
| ![Minimize](icon) | Minimize the Outline view. |
| ![Maximize](icon) | Maximize the Outline view. |
See also
Using Quick Outline on page 400
Opening included files on page 400
Using Filter Preferences on page 402

Going to a specific line number
To go to a specific line in the currently active file, press CTRL+L and enter the line number.

To display line numbers in the ABL Editor, choose Window > Preferences > General > Editors > Text Editors and select the Show line numbers option.

Note: For more information on the navigation features in the Workbench, see the Navigate menu topic in the Workbench User Guide. Also see Setting Eclipse editor preferences

Opening include files
There are several methods for opening include files in the ABL Editor:

• Double-click the include file name in the Outline view.
• Click the include file name in Quick Outline.
• Right-click the include file preprocessor in the code and select Open Declaration from the context menu.
• Press CTRL and click the include file preprocessor in the code.
• Place the cursor over the include file preprocessor in the code and press F3.
• Right-click in the source file and select Show References from the context menu. Then select the include file in the Show References dialog and click Open.

See also
Using the Outline view on page 399
Displaying Quick Outline on page 400

Using Quick Outline
Quick Outline is a pop-up window that shows a tree view of the ABL code file that currently has focus in the ABL Editor.

See also
Using the Outline view on page 399

Displaying Quick Outline
Quick Outline is a pop-up window that shows a tree view of the ABL code file which is currently open in the ABL Editor. You can click on a node in the tree view to navigate to a particular section of the file. It similar to the Outline view, but it is more convenient to work with.

To display Quick Outline:
1. Click anywhere in a file currently open in the ABL Editor.

2. Do one of the following:
   - From the main menu, select **Navigate > Quick Outline**.
   - Press **CTRL+O**.
   - Right-click in the source file, and select **Quick Outline** from the context menu.

3. Click anywhere outside the window or press ESC to close the **Quick Outline** window.

   **Note:** **Quick Outline** toolbar contains (Collapse All) and (drop-down menu) icons. Use the **Collapse All** icon to collapse all expanded tree nodes in the window.

4. The drop-down icon on the **Quick Outline** window contains the following options:

<table>
<thead>
<tr>
<th>Move</th>
<th>Use this to move the position of the <strong>Quick Outline</strong> window. Select <strong>Move</strong>, drag the window to the required position on your screen, and click to release.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resize</td>
<td>Use this to resize the <strong>Quick Outline</strong> window. Select <strong>Resize</strong>, drag the window to resize it, and click to release.</td>
</tr>
<tr>
<td>Remember Size and Location</td>
<td>Use this to capture the size and position settings of the <strong>Quick Outline</strong> window. If you select this option, the <strong>Quick Outline</strong> window is placed at the same position the next time you open it.</td>
</tr>
<tr>
<td>Filters</td>
<td>Use this to show only selected element types in the <strong>Quick Outline</strong> window. See <strong>Using Filter Preferences</strong> on using the filter preferences.</td>
</tr>
<tr>
<td>Sort Ascending</td>
<td>Use this to sort the items in ascending alphabetical order.</td>
</tr>
<tr>
<td>Sort Descending</td>
<td>Use this to sort the items in descending alphabetical order.</td>
</tr>
<tr>
<td>Expand Includes</td>
<td>Use this to display elements in include files.</td>
</tr>
<tr>
<td>Remember Last Filter Text</td>
<td>Use this to view the element types specified in the previous filter settings, the next time you open the <strong>Quick Outline</strong> window.</td>
</tr>
<tr>
<td><strong>Note:</strong> The same filter settings are applied to the <strong>Outline</strong> view.</td>
<td></td>
</tr>
<tr>
<td>Show preprocessors</td>
<td>Use this to display the preprocessors list. The <strong>Quick Outline</strong> is replaced with a table listing preprocessors (with their current values) of the ABL code file which is open in the editor.</td>
</tr>
</tbody>
</table>

**See also**

- Using **Quick Outline** on page 400
- Opening include files on page 400
- Working with preprocessor elements on page 403
Expanding Include files
With Quick Outline open, press **CTRL+I** to display the contents of files included by reference. Press **CTRL+I** again to suppress the display of include files.

See also
Opening include files on page 400
Using Quick Outline on page 400

Showing inherited members
When using Quick Outline in a class file, press **CTRL+O** repeatedly to toggle between showing only members declared in the current class, and showing members inherited from a super class as well. For more information see, Using Quick Outline

Showing preprocessor list
With Quick Outline open, press **CTRL+P** to view the preprocessor list. The outline tree is replaced with a table listing preprocessors (with their current values) of the ABL code file that is open in the editor. The preprocessors values correspond to the current position of the cursor in the ABL Editor. Press **CTRL+P** again to hide the preprocessor list display.

See also
Using Quick Outline on page 400
Listing preprocessors in Quick Outline on page 405

Using Filter Preferences
You can filter the display of the Quick Outline window and the Outline view to show only specified element types, using the Outline Filter Preferences window.

To filter element types from displaying in the view:

1. Do one of the following:

   • From the Outline view toolbar, click View Menu ▼ icon Filters.
   • From the Quick Outline window, click drop-down ▼ icon Filters.

   The Outline Filter Preferences window appears.

2. Do one of the following to select the element types that you do not want to display in the view:

   • Select Name filter patterns checkbox, if you want to filter the element types using patterns. Enter a pattern type in the text field.
   • Select the checkbox against the element type, if you want to filter using the element type. The available element types are:

<table>
<thead>
<tr>
<th>Include file references</th>
<th>Select this to remove all the include file references from the outline.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable definitions</td>
<td>Select this to remove all the variable references from the outline.</td>
</tr>
<tr>
<td>Data members</td>
<td>Select this to remove references to temp-tables, datasets, and buffers from the outline.</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Code sections</td>
<td>Select this to remove references to procedures, functions, methods, and triggers from the outline.</td>
</tr>
<tr>
<td>External references</td>
<td>Select this to remove all code structures that are external to the current program.</td>
</tr>
<tr>
<td>Function and procedure prototypes</td>
<td>Select this to remove all function and procedure prototypes from the outline.</td>
</tr>
<tr>
<td>Using Declarations</td>
<td>Select this to remove all the Using declarations from the outline.</td>
</tr>
<tr>
<td>Inactive preprocessed regions</td>
<td>Select this to remove all the nodes for inactive preprocessor regions from the outline.</td>
</tr>
</tbody>
</table>

3. Click **OK** to save your changes.

**See also**

- [Using the Outline view](#) on page 399
- [Using Quick Outline](#) on page 400

## Working with preprocessor elements

### Enabling preprocessor evaluation

You can enable or disable the **preprocessor** option. If the preprocessing is on, the ABL Editor status bar displays the status as Preprocessing on; otherwise, it displays the status as Preprocessing off. This option is available in all OpenEdge perspectives.

By default the **preprocessor** option is enabled meaning that preprocessors are evaluated. The inactive preprocessor regions in the code are displayed in gray in the ABL Editor.

**Note:** You can use the **Preprocessor inactive** option in the **Editor Colors options** to make the inactive preprocessor regions appear in a different color.

To enable the preprocessor, do one of the following:

- On the main toolbar, click ![](image). The evaluation is on for a file.

  **Note:** To disable the **preprocessor** option, click ![](image) on the main toolbar.

- From the main menu, select **OpenEdge > Enable Preprocessing**.

Note that when the preprocessor is enabled:

- Content assistance displays proposals for built-in or defined preprocessors. The value of a preprocessor at a selected offset is considered to evaluate the proposals.
• When you hover over a preprocessor reference, the value of the preprocessor appears in a pop-up dialog. See also Using hover help.

• Outline view displays the elements in inactive regions irrespective of preprocessing enabled or disabled. See also Viewing preprocessors in the Outline view.

• Quick Outline displays a list of preprocessors and their values. See also Listing preprocessors in Quick Outline.

See also
Working with preprocessor elements on page 403
Using hover help on page 391
Using Quick Outline on page 400
Using the Outline view on page 399
Code assistance on page 369

Viewing preprocessors in ABL source files

When the preprocessor evaluation is on, inactive preprocessor regions for the file that is currently open are displayed in gray in the ABL Editor. This enables you to evaluate and view the code that is excluded by preprocessor.

Note: You can use the Preprocessor inactive option in the Editor Colors options to specify a different color for the inactive preprocessor regions.

When you hover over a preprocessor reference, the value of the preprocessor appears in a pop-up dialog. For example, in the following illustration, hovering over the preprocessor \{&TABLE\} in the ABL Editor displays the current value of preprocessor as Customer in a popup dialog. If the preprocessor evaluation is not completed at the time of hovering, no value is displayed in the hover pop-up dialog.

![Preprocessor Hover Example]

When the preprocessor evaluation is enabled, ABL Editor's code assistance evaluates the value of preprocessors at selected offset and displays the proposals for the defined preprocessor. If the preprocessor option is disabled, a message appears stating Enable preprocessing to get proposals. For example, in the following illustration, when a preprocessor DB is defined, code assistance after \{&DB\} displays tables from the database.

![Preprocessor Code Assistance]

See also
Working with preprocessor elements on page 403
Using hover help on page 391
Viewing preprocessors in the Outline view

By default, with the preprocessor option on or off, the elements inside the inactive preprocessor regions are visible in the Outline view. These elements appear in gray.

**Note:** You can use the Outline Filter Preferences to hide them from the Outline view.

To view inactive preprocessor region elements:

1. Open Outline view by selecting Window from the main menu, and then Show View > Outline.
2. On the Outline view toolbar, select View Menu Filters. The Outline Filter Preferences window appears.
3. Click to clear the Inactive preprocessed regions check box.

**Note:** To hide the inactive preprocessor elements from the Outline view, select the Inactive preprocessed regions check box.

4. Click OK to save your changes.

**See also**
- Working with preprocessor elements on page 403
- Using the Outline view on page 399

Listing preprocessors in Quick Outline

You can use Quick Outline to view all the active preprocessors (and their values) of the file that is open in the ABL Editor. Preprocessor values change based on the cursor position in the ABL Editor.

To display the list of all active preprocessors using Quick Outline:

1. Click in a source file that is open in the ABL Editor.
2. Open Quick Outline by doing one of the following:
   - Press CTRL+O.
   - Right-click in the source file, and select Quick Outline from the context menu.
   - From the main menu, select Navigate > Quick Outline.
3. Display the preprocessors list by pressing CTRL+P. The outline tree is replaced with a list of preprocessors and their values.

**Note:** Press CTRL+P to hide the preprocessors list and view Quick Outline.

4. Press ESC to close the Quick Outline.

**See also**
- Working with preprocessor elements on page 403
Using Quick Outline on page 400

Using the ABL Preprocessor view

The Preprocessor view displays code that is obtained after all the preprocessors (include files, preprocessor definitions, and logic) are expanded. It is a read-only view. Syntax coloring and highlighting are applied to the code in the Preprocessor view, so it appears the same as the code in the ABL Editor.

The Preprocessor view does not appear in the default OpenEdge Editor perspective.

To access the Preprocessor view, do one of the following:

- From the main menu, select Window > Show View > Preprocessor.
- Right-click in the ABL Editor and select Show In > Preprocessor from the context menu.

The Preprocessor view toolbar contains the following buttons:

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Update" /></td>
<td>Update the file displayed in the Preprocessor view.</td>
</tr>
<tr>
<td><img src="image" alt="Link" /></td>
<td>Link the Preprocessor view with the ABL Editor when focus in the ABL Editor changes to another file. By default linking is off.</td>
</tr>
<tr>
<td><img src="image" alt="Minimize" /></td>
<td>Minimize the Preprocessor view.</td>
</tr>
<tr>
<td><img src="image" alt="Maximize" /></td>
<td>Maximize the Preprocessor view.</td>
</tr>
</tbody>
</table>

For more information, see Working with preprocessor elements

Using annotations

Adding templates for code annotations

You can add your own templates to the library of annotations that are available for insertion into ABL source files via the Add Annotation wizard.

To add an annotation template:

1. Choose Window > Preferences. The Preferences window appears.
2. Choose Progress OpenEdge > Editor > Annotations. The Annotation page appears.
3. Click Add to launch the New Annotation dialog.
4. Fill in the new annotation's name and value.
5. Click OK.

Note: The templates that ship with the Annotation Generator contain two specially coded preprocessors, %FILENAME% and %MODULE%. %FILENAME% expands to the filename including the extension. %MODULE% expands to the name of the directory where the file is stored. These are the only preprocessors the tool can expand. The tool does not support customer-created preprocessors.
**Adding annotations to source files**

Annotations are metadata contained in source files. Use the **Add Annotation** wizard to add annotations to specified files.

To add an annotation:

1. Open the **Add Annotation** wizard by choosing **Source > Add Annotation** from the Progress Developer Studio for OpenEdge main menu.

   **Note:** In a file that is open in the ABL editor, you can right-click and choose **Source > Add Annotation** or **Progress OpenEdge > Add Annotation** from the context menu. You can also right-click in the **Project Explorer** view and choose **Progress OpenEdge > Add Annotation**.

2. Select an annotation template from the combo-box. If you are adding ESB annotations, the first option in the drop-down list, see the next topic for detailed instructions.

3. You can modify the annotation text of the selected annotation in the text area provided below the annotation.

4. Check the files to which you want to add the annotation in the **Available Resources** tree view. You can use the **Filter** to narrow your choices.

   **Note:** Class files are not available in the list of resources if the selected annotation template is ESB - Main. The OpenEdge Adapter for Sonic ESB does not support class files.

5. Click **Finish** to add top-level annotations to the selected files. Click **Next** if you want to add annotations to specific procedures or functions.

6. If you click **Next**, you can select available procedures, user-defined functions, and methods from the **Available Resources** tree view. Click **Finish** to add the annotations.

   **Note:** Select the **Back up selected resources** option to save files prior to applying an annotation. When using this function, you must specify the location of the backup.
Adding Sonic ESB annotations

Sonic Enterprise Service Bus (ESB) annotations identify which procedures or user-defined functions in ABL source code will be exported to the Sonic environment. In other words, ESB annotations allow you to expose ABL procedures and user-defined functions as Sonic services. You generate Sonic invocation (.esboe) files from ABL source files that contain ESB annotations. You can import the ESBOE files into Sonic Workbench in order to create ESB processes.

For more information about exposing ABL applications to Sonic ESB, see *OpenEdge Development: Messaging and ESB* in the Product Documentation section of the Progress Software Developer's Network Web site.

You can add ESB annotations to .p, .i, or .w files. You cannot add ESB annotations to class (.cls) files.

The ABL Editor provides a wizard that helps you add ESB annotations to ABL code files. To use this wizard:

1. Click **Edit** if you want to change the ESB Annotation-Main definition.
2. When you click **Edit**, a dialog appears, which allows you to set:
   - **ESBOE file name** - The name of the ESBOE file. By default, the file name is the name of the ABL code file with a .esboe extension. The .p, .i, or .w extension is stripped away.
   - **Execution mode** - Whether the .r file runs externally or persistently. The default is external, but you must change this to persistent if you intend to add ESB annotations to internal procedures or user-defined functions in the file.
   - **Use return value** - Whether a return string should be used for procedures. By default this option is not selected.
   - **Write dataset before image** - Whether before-image data is written when ProDataSet parameters are serialized as XML. By default this option is not selected. This parameter has no effect on invocations that do not return a ProDataSet.
3. Select a project and open a .p, .i, or .w file.
4. Choose **Source > Add Annotation** from the Progress Developer Studio for OpenEdge main menu.
5. If the focus is in an open file, you can choose **Source > Add Annotation** from the context menu. You can also right-click in the Project Explorer view and choose **Progress OpenEdge > Add Annotation**.
6. In the Available Resources section, select the files where you want to add ESB Annotation-Main. For example: The Add Annotations wizard appears.

7. To save a copy of files before annotations are added, select the backup option and specify the save location. For example:
8. Click **Finish** if you do not intend to add ESB annotations to internal procedures or user-defined functions. The wizard adds the ESB Annotation-Main to the top of all selected files.

9. Click **Next** if you intend to add ESB annotations to internal procedures or user-defined functions.

   **Note:** If you have not changed the Execution Mode in the ESB Annotation-Main definition to persistent, the annotation wizard prompts you to make the change. The execution mode must be persistent for internal procedures and user-defined functions to be accessible.

10. Choose **ESB Annotation-Detail**, which is the appropriate annotation for internal procedures and user-defined functions, and which is selected by default.

11. Click **Edit** if you want to change the ESB Annotation-Detail definition.

   **Note:** The attributes are similar to ESB Annotation-Main. However, you cannot specify the execution mode, and the default ESBOE filename contains the name of the ABL code file plus the procedure name.

12. Choose **ESB Annotation-Main**, which is selected by default. For example:

   ![Select annotation or enter annotation text in editor](image)

   **Note:** The ESB Annotation-Main annotation will be added to the top of selected files in your project. You can also add ESB annotations to internal procedures and user-defined functions, but first you must add ESB Annotation-Main to the files that contain the internal procedures and user-defined functions.

13. In the **Available Resources** section, select the files, internal procedures and user-defined functions where you want to add ESB Annotation-Detail annotation. For example:

   ![Available Resources section](image)

14. Click **Finish**.

   **Note:** The ESB Annotation-Detail annotation is added to the selected internal procedures and user-defined functions.

15. Right-click on the project name in the Project Explorer view to generate ESBOE files:

   **Note:** If you do not want to generate ESBOE files for every file in the project, you can select individual file names in the Project Explorer view. You can also right-click in a file that is open in the ABL Editor.

16. Choose **Progress OpenEdge > Generate Sonic ESB Invocation Files**.
Defining BPM service interface for an OpenEdge procedure

To use a procedure .p file in the BPM process workflow, you must add the service interface definition to it. Defining the service interface refers to defining or adding BPM annotations in the procedure file. You can then use the procedure file in the BPM process workflow.

To define the service interface for a procedure file:

1. Right-click on the procedure file in the Project Explorer view or in the Outline tab or the Procedure Editor tab, and select **Progress OpenEdge > Define Service Interface** from the context menu. The **Define Service Interface** wizard appears.
2. Select **BPM** from the **Definition Mode** drop-down.
3. In the Project Explorer view, the procedure selected in Step 1 on page 410 is automatically checked from the list of procedures in the workspace, and the corresponding internal functions and procedures are listed in the Internal Procedures and Functions area.

   **Note:** You can also select multiple procedures from the workspace and define service interfaces for them

4. In the **Backup** area, select the **Backup selected resources** checkbox to back up the procedure files before defining the service interface.
5. Click **Browse** and specify the backup location, and then click **Next**. The **Edit Annotation** page appears.
6. In the **BPM main annotation** area, click **Edit** to edit the BPM annotation for the procedure file. The **New BPM Annotation** dialog appears.
7. In the **New BPM Annotation** dialog, you can specify the following:
   a) Enter the type of annotation to be added in the **Type** field.
   b) Specify the **BIZOE=FileName**.
   c) Select **EXECUTION MODE** from the drop-down.
   d) Select **USE RETURN VALUE** checkbox, if the OpenEdge procedure file returns a value. A retVal parameter is added to the procedure file to store the return value.
   e) Select **WRITE DATASET BEFORE IMAGE** checkbox, if you want before-image data written to the OpenEdge file when ProDataSet parameters are serialized as XML.
8. Click **OK**.
9. In the **BPM detail annotation** area, click **Edit** to edit the BPM annotation for the internal procedures and functions in the main procedure file. The **New BPM Annotation** dialog appears, perform Step 7 on page 410.

   **Note:** You can edit BPM detail annotation only if any internal procedure or function is selected in Step 3 on page 410.
10. Click **Finish**.
Generating a BPM Invocation file

To use the OpenEdge procedure file in a BPM process, you require the BPM annotations in the procedure file, and then generate a .bizoe file for the procedure file. The .bizoe file is required for the procedure file to communicate with the BPM process.

Note: When you drag and drop a procedure file onto the BPM Process workflow (.spt), and then build the BPM process, the .bizoe file is internally generated.

To generate BPM invocation files:

1. Right-click on the procedure file in the Project Explorer view, and select Progress OpenEdge > Generate BPM Invocation Files from the context menu to generate a .bizoe file.
2. Right-click in the Project Explorer tab, and then select Refresh from the context menu. The .bizoe file appears under the project files in the Project Explorer tab.

Note: The generated .bizoe file has the same name as that of the procedure file name. For instance, if you are generating BPM invocation files for a custom.p file, then the generated file will be custom.bizoe. For more information, see Defining BPM service interface for an OpenEdge procedure on page 410.

Working with USING statements

Organizing USING statements

The Organize USING Statements option allows you to organize USING statements in ABL sources through a set of default rules or custom rules that you can define by using a set of preferences. These rules include cleaning, sorting, and grouping USING statements.

Selecting the Organize USING Statements option removes duplicate and unused USING statements and sorts the required ones. It expands .* to a fully qualified package type. It also prompts you if a referenced type name cannot be mapped uniquely to a type in the current project.

Note: The Organize USING Statements option is not available for SpeedScript files. Also, using this option to organize USING statements in a preprocessor directive affects the processing of those USING statements.

To organize USING statements, do the following:

1. Select one or more workspace resources.
2. Select Source > Organize USING Statements from one of the following locations:
   • Main menu bar
   • Project Explorer context menu
   • ABL Editor context menu
   • Outline view context menu
Note: You can also use the keyboard shortcut **Ctrl+Shift+O** to organize USING statements.

If no conflicts are found in the selected resources, Progress Developer Studio for OpenEdge updates the USING statements for the resources.

If you have selected a single resource and if that resource contains conflicting reference types, a dialog is displayed prompting you to select the correct class for the USING statement. If you have selected multiple resources and if at least one of those resources contains conflicting reference types, a dialog is displayed listing the resources with conflicts and prompts you to perform the **Organize USING Statements** operation on each resource separately.

Note:
When you use content assistance (by pressing **CTRL+SPACE**) for selecting a class or interface, the fully-qualified type name is added at the current cursor position and a new or existing USING statement is appended to it based on whether or not a USING statement already exists.

If the operation is not supported and cannot be performed on the selected resources, a message is displayed stating the same.

To undo an **Organize USING Statements** operation, select **Edit > Undo** or press **Ctrl+Z**. To redo, select **Edit > Redo** or press **Ctrl+Y**.

The **Include FROM ASSEMBLY/FROM PROPATH option** preference option on the **USING Statements preferences** page allows you to add a FROM ASSEMBLY or FROM PROPATH option to a USING statement. Adding a FROM ASSEMBLY or FROM PROPATH option to a USING statement narrows the search for the type definition, depending on whether the USING statement specifies an ABL package-based type or a .NET namespace-based type. When you specify the FROM ASSEMBLY option, ABL assumes that the type is a .NET type and uses the list of .NET assemblies defined in the assembly identification file to search for the type definition. When you specify the FROM PROPATH option, ABL assumes that the type is an ABL type and uses PROPATH to search for the type definition. The **Include FROM ASSEMBLY/FROM PROPATH option** preference option is selected by default. For more information, see **USING Statements options**.

See also
- Adding USING statements to ABL resources on page 412
- Specifying keyword casing for USING statements on page 413
- Setting preferences for USING statements on page 413
- USING Statements preferences on page 445
- Source menu on page 446

**Adding USING statements to ABL resources**

Progress Developer Studio for OpenEdge has different rules for adding USING statements to ABL resources with AppBuilder markup and ABL resources without AppBuilder markup. ABL resources with AppBuilder markup include: structured include file, structured procedure, AppBuilder file, and CGI wrapper with AppBuilder markup. ABL resources without AppBuilder markup include: classes, interfaces, procedures, include files, and CGI wrapper without AppBuilder markup.
Follow the below rules to add USING statements to ABL resources with AppBuilder markup:

- If there is an error-handling statement inside the DEFINITIONS block, the location of the error handling statement is used while calculating the insertion position.

- If there is no error-handling statement, USING statements must always be inside &ANALYZE block.

- If &ANALYZE block exists, then the USING statement must be added to this block only.

- If there is no &ANALYZE block, then a new block must be created and appended to the code, and the USING statements must be added inside the newly created &ANALYZE block.

- The newly created &ANALYZE block must be placed after the &ANALYZE-SUSPEND_VERSION-NUMBER block, if it exists. Otherwise, it must be the first block in the result document.

**Note:** The insertion position is also based on the preference options. For more information, see [Setting preferences for USING statements](#) on page 413.

Follow the below rules to add USING statements to ABL resources without AppBuilder markup:

- If there is an error-handling statement, the location of the error handling statement is used while calculating the insertion position.

- If there is no error-handling statement, the first ABL statement is used while calculating the insertion position.

- If there are no existing (child) statements and the first ABL statement starts with type or procedure declarations, then the new USING statements must be added above the type or procedure declaration. In case of classes and interfaces add the USING statements above the type declaration and in case of procedures add them just above the first procedure.

**Note:** The insertion position is also based on the preference options. For more information, see [Setting preferences for USING statements](#) on page 413.

### Specifying keyword casing for USING statements

You can use the Editor preferences page to specify the keyword casing for USING statements.

To specify keyword casing for USING statements:

1. Select **Window > Preferences > Progress OpenEdge > Editor**
2. Select one of the following options on the Editor preferences page:

   - **No action:** Retains the original keyword casing for all the existing USING statements, which remain after the Organize USING Statements operation is performed, and applies uppercase to keywords in the newly added USING statements.
   - **Upper:** Applies uppercase to keywords in all the existing USING statements, which remain after the Organize USING Statements operation is performed, and to keywords in the newly added USING statements.
   - **Lower:** Applies lowercase to keywords in all the existing USING statements, which remain after the Organize USING Statements operation is performed, and to keywords in the newly added USING statements.

### Setting preferences for USING statements

You can control the USING statements’ functionality (position, grouping, etc.) in your current workspace by setting preferences on the USING Statements preferences page.
To set preferences for USING statements:

1. From the main menu, select **Window > Preferences**.
   The **Preferences** page appears.
2. Select **Progress OpenEdge > Advanced > USING Statements**.
   The **USING Statements** preferences page appears.
3. Set the following preferences for the USING statements:
   - Position of the statements
   - Grouping and sorting of the statements
   - Style of the statements
   - Inclusion of the FROM ASSEMBLY/FROM PROPATH option
   - Preservation of valid statements
4. Select **Apply**.
5. Click **OK** to save your settings and close the preferences page.

**Note**: To prevent existing, valid USING statements from being relocated, they must be preserved in their original form and must not be grouped together. For more information, see the **USING Statements preferences page**.

**See also**
- Organizing USING statements on page 411
- Adding USING statements to ABL resources on page 412
- Specifying keyword casing for USING statements on page 413
- USING Statements preferences on page 445

**Compiling, testing, and running ABL programs**

**Checking syntax**

To check the syntax of the active file in the ABL Editor, press **CTRL+SHIFT+C**, or right-click and select **Check Syntax** from the context menu.

**See also**
- Clearing OpenEdge errors on page 417
- Compiling ABL code on page 414
- Using assistance features on page 389

**Compiling ABL code**

By default, Eclipse compiles source files when they are added to the workspace or saved. This is the case unless you disable the Build automatically option in Eclipse Workspace preferences (**Window > Preferences > General > Workspace**).
You might choose to disable automatic builds if the contents of your workspace are frequently updated significantly (for example, by copying files from an external source), triggering build processes that block creating or saving resources until they finish. Otherwise, you should enable automatic builds to ensure that up-to-date r-code is always available for tools and features that require it.

If you do disable automatic builds, it is recommended that you have Progress Developer Studio for OpenEdge compile ABL source files when you save them. To do so, select the Compile on save if required option in Editor Build preferences.

At any time, you can explicitly compile your current file by selecting Compile from either the Source menu or the ABL Editor context (right-click) menu.

Startup parameters for compiler operation

Progress Developer Studio for OpenEdge provides two startup parameters that may affect the way the compiler processes source code: the Compile Warning List (-cwl) and the Keyword Forget List (-k). These options are briefly described here. For more information, see OpenEdge Deployment: Startup Command and Parameter Reference.

Compile Warning List (-cwl)

The Compile Warning List specifies a set of ABL statements that trigger a warning message at compile time if they are found in the source code. The list can include any ABL statements you choose; it can be an effective way to discourage the use of deprecated statements. The occurrence of warnings does not prevent valid code from compiling.

Each warning appears in the following format:

The flagged_statement statement from the compile warning list was found in file file_name at line line_number.

If you see one or more such messages when you compile your code, click OK to continue, and if necessary, edit the code as appropriate.

Keyword Forget List (-k)

The Keyword Forget List specifies a set of ABL keywords that the compiler should not treat as keywords when processing the source code. This option allows you to avoid compilation errors that might otherwise occur if an upgraded version of Progress Developer Studio for OpenEdge introduces new keywords that are found in existing code as object names (such as tables, fields, frames, variables, streams, and so on).

If you encounter errors of this sort, you can use a Keyword Forget List to disable selected keywords, and thereby allow the code to compile, until you can remove those words from the application.

See also

Going to a specific line number on page 400
Running and debugging programs on page 417
Setting ABL Editor preferences on page 379
Editor Build options on page 438

Setting Compile options preferences

You can set the preferences for Compile options on the workspace level or the project level. If you want to set the default preferences for all the projects, instead of manually configuring each project, you should set the preferences at the workspace level. However, you can override the workspace preference for a particular project by setting the project properties.

To set the preferences at the workspace level, click Windows > Preferences > Progress OpenEdge > Compile.
To set the project properties, right-click a project and click Properties > Progress OpenEdge > Build > Compile.

**Note:** If you update any compile options in project properties, the options specified in the workspace are overwritten on the Preference page.

1. Select the **Save r-code (also enables include file tracking)** check box to save .x files when you build the code. When this option is enabled, OpenEdge saves compiled .x files in the directory you specify in the **Build destination** field; or, if that field is blank, in the same directory as the source file.

2. Enter a key for encrypted source files in the **XCODE key** field. Leave this field blank if you do not use encrypted source code.

3. Specify the procedure that runs prior to compilation in the **Pre-compile callback routine** field.

4. Select the **Create cross-reference files in XML format** check box to save cross references in an XML file. This corresponds to the COMPILe option, XREF-XML.

**Note:**
Cross-referenced objects include procedure and include files, user-defined functions, classes, methods, tables, fields, variables, frames, and character strings. For more information about XREF-XML and other COMPILe options, see the ABL Language Reference.

5. Specify the path of the folder where cross references are saved in the **XREF XML destination** field.

6. Select the following check boxes as required:
   - **Require full names:** Specifies that all table and field names must be their full unabbreviated names as they appear in the schema. If you select this check box, it disables the compiler’s ability to resolve abbreviated names in the tables.
   - **Require field qualifiers:** Specifies that all buffer references (database tables, temp-tables, and buffers) must be fully qualified. If you select this check box, it disables the compiler’s ability to implicitly resolve the buffer to which a field reference refers to.
   - **Require full keywords:** Specifies that all keywords must be their full unabbreviated names as they appear in the schema. If you select this check box, it disables the compiler’s ability to resolve abbreviated names in the keywords.
   - To enable the project level properties, select the **Enable project-specific settings** check box and select the required compile options.

7. When you click **Apply** after updating the options, the **Compiler settings changed** dialog box opens. Do one of the following:
   - Click **Yes** to invoke the full build of all the projects in the workspace.
   - Click **No** to apply the selected options but the build is not invoked.
   - Click **Cancel** to cancel the changes and the build is not invoked.

**Setting breakpoints while editing**
You can set or remove breakpoints in your code for use in testing and debugging:
1. Position the cursor in the line where you want to add or remove a breakpoint.
2. Right-click and choose Toggle Breakpoint.

**Note:** Alternatively, double-click in the marker bar (the left margin) at the line where you want to insert or remove a breakpoint.

If the selected line was not already marked as a breakpoint, a blue circle \( \circ \) appears in the left margin, indicating a breakpoint. If the line was previously marked, the breakpoint is removed, and the blue circle disappears.

**See also**  
Running and debugging programs on page 417

**Clearing OpenEdge errors**

You can clear any Progress Developer Studio for OpenEdge error messages for a particular file from the Problems view or the marker bar by choosing **Progress OpenEdge > Clear OpenEdge Compile Errors** from the ABL Editor context (right-click) menu.

The **Clear OpenEdge Compile Errors** command does not correct the errors. Any unfixed errors are flagged again when you next compile the code.

**See also**  
Checking syntax on page 414  
Compiling ABL code on page 414  
Running and debugging programs on page 417

**Running and debugging programs**

Running and debugging programs in the Eclipse environment involves the use of launch configurations. Detailed instructions for defining and using launch configurations, as well as for using the Progress Developer Studio for OpenEdge Debugger, are provided in the Progress Developer Studio for OpenEdge help volume "Running and Debugging ABL Programs." Refer to that online help to gain a clear understanding of the relevant concepts and procedures.

Learning more about running and debugging is desirable, but in general, you can:

- Run the currently selected program by clicking **Run** \( \mathfrak{r} \) on the toolbar.
- Launch the Progress Developer Studio for OpenEdge Debugger and debug the current program by clicking **Run** \( \mathfrak{r} \) on the toolbar.

**See also**  
Compiling, testing, and running ABL programs on page 414
Excluding resources from the build

You can exclude compilable resources when you build your project. Excluding resources option is available only for the resources listed in the Source directory viewer on the Source tab of the Build properties page. For example, if the Source directory viewer displays ROOT as the entry, the Exclude from Build option is enabled for all the compilable resources (.p, .cls, .w, or .htm files, for example) under the project’s root folder.

**Note:** See Editor Build options for the list of compilable file extensions defined for your project.

You can exclude resources using the Build path or the Source tab of the Build properties page. When excluding the resources, note the following:

- Excluding a folder also excludes all of its files and folders from the build.
- You can individually select a file or a folder to exclude from the build.
- When you exclude resources (files or folders) from a folder, they remain excluded from the build even if the folder is excluded from the build, and then included at a later point. For example, if you exclude a file called customer.cls under the folder called SampleApp from the build, then exclude the folder SampleApp from the build, and then include the folder SampleApp, then the customer.cls file remains excluded from the build.

Excluding resources from the context menu

Exclude from Build allows you to exclude compilable resources when you build your project. You can exclude files, folders, or both from the build.

To exclude resources using Exclude from Build:

1. Open an OpenEdge project.
2. In the Project Explorer view, select the resources that you want to exclude from the build.
3. Right-click the selected resources and select Progress OpenEdge > Exclude from Build from the context menu.

**Note:** The context menu displays either Exclude from Build or Include in Build depending on whether a resource is included or excluded. The context menu option is disabled when there is a combination of excluded or included resources, there exists a non-compilable resource in the selection list, or the selected resource is not part of the Source tab entries.

The selected resources are now excluded from the build, and the compiler ignores them when building your project. You can view the excluded resources in the Source tab of the Build properties page.

See also

- Excluding resources from the build on page 418
- Including resources in the build on page 420
- Compiling, testing, and running ABL programs on page 414
- Building resources using working sets on page 421

Excluding resources using Exclusion Pattern Selection

To exclude resources using Exclusion Pattern Selection:
1. Open an OpenEdge project.
2. Open the project properties page by doing one of the following:
   • Selecting a project, and clicking **ALT+ENTER**.  
   • Right-clicking a project, and selecting **Properties** from the context menu.
3. Access the **Excluded** node by doing one of the following:
   • Selecting **Progress OpenEdge > Build > Source tab > Root** node of the project.
   • Selecting **Progress OpenEdge > PROPATH > Root** node of the project.
4. Select the **Excluded** node, and click **Edit**. The **Exclusion Rules Wizard** appears.

   **Note:** The **Edit** option is enabled only when you select the **Excluded** node.

5. Click **Add Pattern**. The **Add Exclusion Pattern** appears.
6. In the **Exclusion Pattern** field, enter a file pattern for exclusion. The build looks for the resources that match the file pattern defined, and excludes them from the build. If you want to exclude all the resources of the same file extension, you can use * or ?. For example, to exclude resources of the same file extension as `.cls`, use the pattern: `/SampleApp/openedge/tutorial/UI/*.cls`.
7. Click **OK** to save your changes.

   The resources excluded from the build appear in the **Excluded** node, and are ignored when your project is built.

**See also**

- **Excluding resources from the build** on page 418
- **Including resources in the build** on page 420
- **Compiling, testing, and running ABL programs** on page 414
- **Building resources using working sets** on page 421

**Excluding resources using Add Resource Exclusions**

To exclude resources using **Add Resource Exclusions**:

1. Open an OpenEdge project.
2. Open the project properties by doing one of the following:
   • Selecting a project, and clicking **ALT+ENTER**.  
   • Right-clicking a project, and selecting **Properties** from the context menu.
3. Access the **Excluded** node by doing one of the following:
   • Selecting **Progress OpenEdge > Build > Source tab > Root** node of the project.
   • Selecting **Progress OpenEdge > PROPATH > Root** node of the project with the **Excluded** node.
4. Select the **Excluded** node, and click **Edit**. The **Exclusion Rules** wizard appears.

   **Note:** The **Edit** option is enabled only when you select the **Excluded** node.
5. Click **Add Resources**. The **Add Resource Exclusion** appears.

6. Select the files or folders that you want to exclude from the build.

7. Click **OK** to save your changes.

   The resources excluded from the build appear in the **Excluded** node, and are ignored when your project is built.

**See also**

- Excluding resources from the build on page 418
- Including resources in the build on page 420
- Compiling, testing, and running ABL programs on page 414
- Building resources using working sets on page 421

**Including resources in the build**

The resources (files or folders) that were excluded from the build can later be included in the build. You can include resources in the build using either the **Build** path or the **Source** tab of the **Build properties** page. When including the resources in the build, note the following:

- Including a folder also includes all of its files or subfolders in the build.
- You can individually select a file or a folder to include in the build.
- You cannot include specific files or folders in the build from an excluded folder instead you must include the top folder to include all of its files or folders in the build.

**Including resources using the context menu**

You can include the excluded resources (files or folders) to the build using the **Include in Build** option. This option is only available for the files or folders that were excluded from the build.

To include resources using **Include in Build**:

1. Open an OpenEdge project.
2. In the Project Explorer view, select the excluded files or folders that you want to include in the build.
3. Right-click the selected resources and select **Progress OpenEdge > Include in Build** from the context menu.

**Note:** The context menu displays either **Exclude from Build** or **Include from Build** depending on whether the resources are either included or excluded. The context menu option is disabled when some resources are excluded and some are included.

The selected resources are now included in the build, and are compiled.

**See also**

- Including resources in the build on page 420
- Adding SQL connection profiles on page 418
- Building resources using working sets on page 421
- Compiling, testing, and running ABL programs on page 414
Including resources using the Source tab

You can include the excluded resources (files or folders) to the build using the Source tab of the Build properties page. This option is only available for the files or folders that were excluded from the build.

To include resources using the Source tab:

1. Open an OpenEdge project.
2. Open the project properties by doing one of the following:
   - Selecting the project, and clicking ALT+ENTER.
   - Right-clicking the project, and selecting Properties from the context menu.
3. Access the Excluded node by doing one of the following:
   - Selecting Progress OpenEdge > Build > Source tab > Root node of the project.
   - Selecting Progress OpenEdge > PROPATH > Root node of the project with the Excluded node.
4. Include the resources that were excluded from the build by doing one of the following:
   - If you want to include all the resources that were excluded from the build, select the Excluded node, and click Remove. Go to Step 7 on page 421.
   - If you want to include selective resources, select the Excluded node, and click Edit. The Exclusion Rules appears. Go to Step 5 on page 421.

Note: The Edit option is enabled only when you select the Excluded node.

5. In Exclusion Rules, select the resources that you want to include in the build from the Manage do-not-compile list section, and click Remove.
6. Click Finish to save your changes.
7. Click OK to close the project properties page.
   The resources are included in the project build, and the corresponding r-code files for these resources are generated when your project is built the next time.

See also
Including resources in the build on page 420
Excluding resources from the build on page 418
Building resources using working sets on page 421
Compiling, testing, and running ABL programs on page 414

Building resources using working sets

You can use working sets to group resources so that you can work on a set of resources instead of working with the entire project. When you select a working set, only the resources contained in the defined set appear in the Project Explorer view.
You can use a working set to build a set of resources of your project, instead of building the entire project. To build resources using a working set, you must first define a working set with the resources that you want to include in the build.

**Note:** Resources excluded from the build are not considered during the build, even if you add these resources to a working set.

### Creating a new working set

The Select Working Set window allows you to create a new working set to work or build a set of resources of your project.

To create a new working set:

1. Open an OpenEdge project.
2. Do one of the following:
   - On the Project Explorer toolbar, click View Menu, and then Select Working Set.
   - On the main menu, select Project > Build Working Set > Select Working Set.

**Note:** The Select Working Set option under Build Working Set is enabled only when the automatic build preference (at the project level) is disabled (ensure that Project menu > Build Automatically is not selected).

The Select Working Set window appears.

3. Click New. New Working Set appears.
4. From the Working set type list, select Resources.
5. Click Next.
6. In the Working set name field, enter a name for the new working set.
7. Add files and folders to the new working set by doing one of the following:
   - If you want to add individual file and folder, select the check box for the file or folder in the Working set contents section.
   - If you want to add all the available files and folders, click Select All.

**Note:** You can use the Deselect All option to remove all the resources added to the working set.

8. Click Finish to save your changes.

This creates a working set with the set of resources that you have selected. You can now work or build only the set of resources defined in the working set instead of the entire project.

**See also**

- Resources excluded from the build on page 367
- Building resources using working sets on page 421
- Excluding resources from the build on page 418
- Including resources in the build on page 420
Building a working set automatically

By default, builds are performed automatically when you save resources.

To build a working set automatically:

1. Open an OpenEdge project.
2. View the working sets defined in your project by doing the following:
   - On the Project Explorer toolbar, click View Menu.
   - Select Top Level Elements > Working Sets.

   The working sets defined for your project appear in the Project Explorer view.
3. Select a working set that you want to build automatically.

   Note: You can select more than one working set to build automatically.
4. From the main menu, select Project > Build Working Set Automatically.

   Note: The Build Working Set Automatically option is enabled only when the automatic build preference (at the project level) is disabled (ensure that Project > Build Working Set Automatically is not selected).

When you save resources, the selected working set(s) are built. If the working set contains files or folders from different projects, the corresponding project ABL Virtual Machines (AVMs) are used for building the selected working set(s).

See also
- Resources excluded from build on page 367
- Creating a new working set on page 422
- Building resources using working sets on page 421
- Excluding resources from the build on page 418
- Including resources in the build on page 420
- Compiling, testing, and running ABL programs on page 414

Building a working set manually

If you need more control when builds occur of your working sets, you can disable automatic building and then manually invoke builds.

Note: You can build only one working set at a time when you manually invoke builds.

To manually build a working set:

1. Open an OpenEdge project.
2. On the main menu, select Project > Build Working Set.
Note: The working sets under Build Working Set are only available when the automatic build preference is disabled (ensure Project > Build Automatically is not selected).

3. From the list of available working sets, select a working set that you want to build.
   The selected working set is manually built.

See also
- Resources excluded from build on page 367
- Creating a new working set on page 422
- Building resources using working sets on page 421
- Excluding resources from the build on page 418
- Including resources in the build on page 420
- Compiling, testing, and running ABL programs on page 414

Finding references to ABL constructs

To find references to the ABL constructs, right-click the ABL source (.p, .i, .w, or .cls file) node from the Project Explorer view, select References, and Workspace or Project depending on where you want to search for the references.

Note: You can also select References by right-clicking the ABL element in ABL source file in the ABL source editor or right-clicking the ABL source (or the context menu actions) in the Outline view. You can also select the code in the editor and press the Ctrl+Shift+G key sequence to search references in the workspace.

The result appears in the Eclipse Search view.

If you select an invalid ABL element for example USING statements as in the example below, no result is displayed:

```
USING Batching.someform.
```

Using the Search view

The Search view displays the search results of ABL construct references in an ABL file, project, or workspace. You can view it in the following modes:

- **Tree mode**: This is the default mode of the Search view and is similar to the Project Explorer view. The references found in each file are grouped and displayed under the first code block (method or procedure). Results with more than one matching reference are grouped and displayed as a single node, instead of showing a different node for each matching reference, and the total number of matching references in that node is displayed in the label.

- **List mode**: To change the Search view mode to a list mode, select the drop-down arrow on the view’s toolbar and select Show as List. In this mode, all the matching references are displayed in a list form. The label displays the file in which the match was found, separated with a hyphen.

The Search view toolbar has the following controls:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show Next Match</td>
<td>Displays the next matching reference in the view.</td>
</tr>
<tr>
<td>Control</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>Show Previous Match</td>
<td>Displays the previous matching reference in the view.</td>
</tr>
<tr>
<td>Show Previous Searches</td>
<td>Displays the previous search results.</td>
</tr>
<tr>
<td>Remove</td>
<td>Deletes the selected entry from the view.</td>
</tr>
<tr>
<td>Remove All</td>
<td>Deletes all the entries from the view.</td>
</tr>
<tr>
<td>Collapse All</td>
<td>Collapses all the currently expanded entries and displays only the top-level data elements.</td>
</tr>
<tr>
<td>Expand All</td>
<td>Expands all the top-level data elements in the view.</td>
</tr>
<tr>
<td>Search Again</td>
<td>Starts a new search process.</td>
</tr>
</tbody>
</table>

**Generating ABLDoc documentation**

**Generating the ABLDoc documentation using the Generate ABLDoc wizard**

You can generate the ABLDoc documentation from the Generate ABLDoc wizard.

To generate the ABLDoc documentation:

1. Create an OpenEdge project or select an existing one.
2. From the project or editor's context-menu, select **Progress OpenEdge > Generate ABLDoc**. The Generate ABLDoc wizard appears.
3. In the **Document title** field, specify a name as the title for the generated ABLDoc documentation.
4. In the **Destination** field, add or browse for a default directory to save all output files.
5. In the **Select the files for which you want to generate the ABLDoc** field, select the project and the corresponding files for which you want to generate the ABLDoc documentation.

**Note:** The left section displays the projects and their directory structure and the right section displays the ABL elements in the selected directory.

6. In the **Create ABLDoc for members with access mode** field, select Private or Protected depending upon if you want to include members with visibility other than public. Members with Public access only are displayed by default.
7. Click **Finish**.

The ABLDoc documentation file is created in the specified location for the selected project. If another file with the same name already exists in that location, the file is overwritten.
Generating ABLDoc using Apache Ant

You can use a custom ABLDoc Ant task to generate ABLDoc documentation for ABL sources. The ABLDoc Ant task is part of ant-abldoc.jar and is located at:

%DLC%\java

To generate ABLDoc using the ABLDoc Ant task, ensure that Apache Ant is installed on your machine and then copy the ant-abldoc.jar file to the installation location of Apache Ant.

To run the ABLDoc Ant task, you must meet the following prerequisites:

- Configure the classpath to include the following JAR files:
  - abldoc-core.jar
  - ast.jar ast-dependencies.jar
  - velocity-version.jar (version must specify the version number; for example 1.7))
  - velocity-dep-version.jar (version must specify the version number; for example 1.7)

  These JAR files are available at:
  %DLC%\oeide\eclipse\plugins\com.progress.openedge.pdt.abldoc.core._version\lib
  (where version must specify the version number; for example: 11.5.0.00).

- Specify the default location of the required artifacts (which include, style sheets (.css files), images, and velocity templates.

  These artifacts are available at:
  %DLC%\oeide\eclipse\plugins\com.progress.openedge.pdt.abldoc.core._version\abldoc-artifacts
  (where version must specify the version number; for example: 11.5.0.00).

Note: The directory structure of abldoc-artifacts must not change.

ABLDoc task-level properties

You can set the following properties at the ABLDoc Ant task level:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dlc</td>
<td>Specifies the location of your OpenEdge installation.</td>
</tr>
<tr>
<td>destdir</td>
<td>(Optional) Specifies the destination directory where the ABLDoc documentation is generated. If not specified, the current working directory is used as the default destination directory.</td>
</tr>
<tr>
<td>doctitle</td>
<td>(Optional) Specifies the documentation title. If not specified, ABLDoc Documentation is used as the default title.</td>
</tr>
<tr>
<td>verbose</td>
<td>(Optional) Enables or disables the verbose mode. The valid values are true, false, on, off, yes, and no. By default, the value is set to false; and the verbose mode is off.</td>
</tr>
</tbody>
</table>
**Tasks**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>includeExtension</td>
<td>(Optional) Specifies a comma separated list of include file extensions. If not specified, i is used as the default file extension.</td>
</tr>
<tr>
<td>artifactLoc</td>
<td>Specifies the location of the required artifacts.</td>
</tr>
<tr>
<td>serializer</td>
<td>(Optional) Specifies the serializer used by the ABLDoc documentation generator. Also, lets you provide a custom serializer. <strong>Note:</strong> If you are using a custom serializer, you must implement the <code>com.progress.openedge.pdt.abldoc.core.serializer.IdocSerializer</code> interface. This interface is part of the <code>abldoc-core.jar</code> file. If not specified, HTML serializer is used as the default serializer.</td>
</tr>
<tr>
<td>tagParser</td>
<td>(Optional) Lets you provide a custom tag parser to parse the comments. <strong>Note:</strong> To use a custom tag parser, you must implement the <code>com.progress.openedge.pdt.abldoc.core.tag.ITagParser</code> interface. This interface is part of the <code>abldoc-core.jar</code> file.</td>
</tr>
</tbody>
</table>

**ABLDoc elements**

You can use the following ABLDoc element in the ABLDoc Ant task.

**abldocproject element**

Represents a set of resources required to generate ABLDoc documentation. Each ABLDoc Ant task can be configured with one more abldocproject elements.

Set the following properties for the `abldocproject` element:
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>(Optional) Specifies the name of the project. It is only used to indicate failure to generate ABLDoc.</td>
</tr>
<tr>
<td>sourcepath</td>
<td>Specifies a comma separated list of paths that can be configured with each project where the ABL source files can be found. The <strong>sourcepath</strong> element contains a set of ABL source files for which the ABLDoc documentation is generated. It can be configured with Ant’s <strong>pathelement</strong> element or <strong>filesset</strong> element.</td>
</tr>
</tbody>
</table>

**Note:** The **sourcepath** element can also be added as an element to the root **abldocproject** element.
Sample ABLDoc Ant task

Below is a sample build.xml file (ABLDoc Ant task):

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project basedir="." default="main" name="ABLDoc-Ant">
 <!-- Properties used to configure the CLASSPATH and the default artifacts location -->
 <property name="DLC_HOME" value="C:/Progress/OpenEdge115"/>
 <property name="version" value="11.5.0.00"/>
 <property name="PROJECT_HOME" value="${DLC_HOME}/oeide/eclipse/plugins/com.progress.openedge.pdt.abldoc.core_${version}"/>
 <property name="lib.dir" value="${PROJECT_HOME}/lib"/>
 <property name="artifacts.dir" value="${PROJECT_HOME}/abldoc-artifacts"/>
 <!-- Target for defining 'taskdef' -->
 <target name="taskdef">
   <taskdef name="abldoc" classpath="${DLC_HOME}/java/ant-abldoc.jar" classname="com.progress.openedge.ant.abldoc.ABLDocTask">
     <!-- Class path for including the dependencies -->
     <classpath>
       <fileset dir="${lib.dir}">
         <include name="**/*.jar"/>
       </fileset>
     </classpath>
   </taskdef>
 </target>
 <!-- main task to generate the documentation -->
 <target name="main" depends="taskdef">
   <abldoc includeProtected="true" includePrivate="true" tagParser="com.progress.abldoc.CustomTagParser" serializer="com.progress.abldoc.CustomSerializer" artifactLoc="${artifacts.dir}" includeExtension="i" verbose="yes" doctitle="ABLDoc Documentation" destdir="${basedir}/docs" dlc="${DLC_HOME}">
     <abldocproject name="sample-prj" sourcepath="C:/abldoc/sample-prj" propath="C:/abldoc/lib" location="C:/abldoc/sample-prj"/>
     <abldocproject name="project 2" location="C:/abldoc/sample-prj1"/>
     <pathelement location="C:/abldoc/lib"/>
     <sourcepath>
       <pathelement location="C:/abldoc/sample-prj1/ pros"/>
       <fileset dir="C:/abldoc/sample-prj1/classes">
         <include name="**/*.cls"/>
       </fileset>
     </sourcepath>
   </abldoc>
 </target>
</project>
```

Navigating through the ABLDoc Navigation page

The ABLDoc documentation that you generate in HTML format (default) is located in the output directory that you specified in the Generate ABLDoc wizard.

To navigate through the ABLDoc navigation page:

1. Double-click the index.html file. The ABLDoc navigation page opens. The document title appears on the top and is what you specified in the Generate ABLDoc wizard.

2. Be default, tree structure to list the packages in the project and their folder structure is displayed; click the tree structure toggle button to hide or show the list.
Adding comments

You can add comments to your source files so that the comments are published to the ABLDoc document. While adding comments, you can use HTML tags as follows to structure the comments in the generated output:

```html
/*--------
File     : <h1><i>GenericDataExporter</i></h1>
Purpose  :
Syntax   :
Description :
Author(s) : abc
Created   : <pre>Tue Nov 19 21:16:10 CET</pre>
Notes     :
------------------------------------------------------------------------*/
```

The comment can have a tag section. You can also add tags to comments to add metadata. These tags are applicable only to methods, functions, and internal procedures. Place the tags after the main description. You can add the following tags:

- **@param**: Adds a parameter with a specified name and description.
- **@return**: Adds a return section to the document with the description that you provide.

Adding comments to the Overview page:

Each application or set of packages can have an overview comment that applies to the application or all the packages, for example, a description of an application.

1. Open the Overview_template.vmin in the
   `DLC\oeide\eclipse\plugins\com.progress.openedge.pdt.abldoc.core._version\abldoc-artifacts/templates` location.
2. Add your comments to this file. When you generate the ABLDoc documentation, the comment appears at the top of the Overview page.

Adding comments to a package

You can add comments to each package in the ABL source code. Each package should have its own .html file for comments. A comment appears as the first sentence on the package summary page.
1. Create an HTML file, assign a name to it (for example package.html), and place the file in the folder that contains the package.

2. Add your comments to the file. A comment should be wrapped in the <body> tag as follows:

```html
<HTML>
<BODY>
This comment is used to describe the sample package,
The contents of this <body> tag will be placed in the summary page,

</BODY>
</HTML>
```

Adding comments to a file

You can also add comments to a file in the ABL source code by adding comments to the Purpose section in the following format:

```ABL
/*------------------------------------------------------------------------
File   : GenericDataExporter
Purpose : 
Syntax  :
Description :
Author(s)  : abc
Created    : Tue Nov 19 21:16:10 CET 2013
Notes     :
------------------------------------------------------------------------*/

USING Progress.Lang.*.

BLOCK-LEVEL ON ERROR UNDO, THROW.

CLASS com.ClassA:
END CLASS.
```

Adding comments to elements

You can also add comments for elements in the ABL source code. Add the comment just above the declaration of the element in the following format:

```ABL
/*------------------------------------------------------------------------------
Purpose:Picks up this comment
Notes:  
@param this is an integer param
@return
------------------------------------------------------------------------*/

METHOD PUBLIC VOID main( a as int ):
    RETURN.
END METHOD.
```

Customizing ABLDoc output

By default, you can generate an ABLDoc documentation in HTML format for your ABL source code. However, you can customize the output for other formats for example, PDF, or customize the look for the HTML output.

Customizing templates and .css files

You can customize the default HTML output for your source ABL code by customizing templates. The location of the ABLDoc artifacts is:
The HTML files are generated using templates in the %artifact-location%/templates location. You can customize the following templates:

- **Navigation_template.vm** – To customize the output of the main navigation page.
- **Overview_template.vm** - To customize the Overview page. You can also add a detailed description of the application to this template.
- **Class_template.vm** - To customize the output of the class files.
- **Procedure_template.vm**: To customize the output of the procedure files.
- **Fields_summary_template.vm**: To customize the structure of Fields Summary section of class and procedure files.
- **Fields_detail_template.vm**: To customize the structure of Fields Detail section for classes and procedures.
- **Copyright_template.vm**: Placeholder for copyright information. The information placed in this template is displayed in all pages.

You can customize the .css file and images or add a new .css file in the following location:

```
%artifact-location%/resources/css
%artifact-location%/resources/images
```

**Note:** All the files or folders in the resources folder are copied the output destination directory. If you want any images or source code to be copied to the output destination directory, you can place them in the resources folder.

### Adding a custom serializer class

To generate a custom format, you can add a custom serializer to the ABLDoc framework. You can use the ABLDoc model to get the file information and serialize it to the format you want. When output is generated, your custom serializer class is called and not the default HTML serializer.

1. Copy the abldoc-core.jar file from
   
   `{DLC_HOME}\oeide\eclipse\plugins\com.progress.openedge.pdt.abldoc.core_11.5.0.00\lib
   
   into an Eclipse project or a folder.

2. Create a class that implements the IDocSerializer interface (available in the abldoc-core.jar file)

3. Override the generateDoc methods and add your logic for generating the custom format.

   **Note:** The generateDoc method has an argument RootNode that has all the packages and files. All the children of this node can be viewed using NodeVisitor.

4. Compile the class with abldoc-core.jar in the classpath.
Note:
To add a custom serializer from Progress Developer Studio for OpenEdge, go to
DLC_HOME\oeide\eclipse\plugins\com.progress.openedge.pdt.abldoc.ui, and edit abldoc.properties to
add new properties as follows:

serializer: fully qualified name of the class
serializerpath: the absolute path of the folder where this class is located. If you generate the ABLDoc
documentation, the custom serializer that is added, is called.

Customizing tag parser
ABLDoc understands the default comment convention (as discussed in the Adding comments on page 430
section), if you want to change the format of the comments, you can add a custom tag parser to parse the
comments.

1. Copy the abldoc-core.jar file from
   (DLC_HOME)\oeide\eclipse\plugins\com.progress.openedge.pdt.abldoc.core_11.5.0.00\lib
   into an Eclipse project or a folder.
2. Create a class that implements the ITagParser interface (available in the abldoc-core.jar file)
3. Override the getHandleTypes and populateTags methods.

Note: getHandleTypes method should return the type of object comments (like methods or procedures)
that is to be parsed using this parser. All the supported types are available in the DocType enum. The
populateTags method is called for all the objects that you want to handle (the types returned by the
getHandleTypes); this method is called back with the IDocModel parameter. It has all the information about
the program elements such as raw comment parameters.

4. Parse the comment and populate the tag objects (com.progress.openedge.pdt.abldoc.core.model.Tag) in
   the IDocModel.
   You can use the populated tags in the templates or custom serializer.

Reference
ABL Editor preferences

Editor options
The Editor page (Progress OpenEdge > Editor) of the Editor preferences includes the following options:
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| Configure tab-space preference                  | Links to the Eclipse Text Editor preference page, where you can specify tab behavior and other options that affect the ABL Editor (and all other text editors running in the Eclipse environment). Two Eclipse settings affect tab behavior:  
  • Displayed tab width sets the number of spaces between tab stops. Note that in all cases, pressing TAB moves the cursor a variable number of spaces to the next tab stop; this number depends on the starting position of the cursor relative to the tab stops.  
  • Insert spaces for tabs determines whether white space inserted before a tab stop consists of individual space characters or a single tab character. |
| Replace content when tab is pressed in overwrite mode | When Insert spaces for tabs is selected in Eclipse Text Editor preferences, determines whether pressing TAB in overwrite mode replaces (with spaces) existing characters between the starting cursor position and the next tab stop. If this option is turned off, the tab spacing is inserted, leaving existing characters unchanged. (When Insert spaces for tabs is not selected, pressing TAB always inserts tab spacing, whether in insert mode or in overwrite mode.) |
| Enable smart indent of new lines                | Determines whether the ABL Editor automatically indents lines as you type for improved readability.                                            |
| Enable auto indent on paste                     | Determines whether code copied and then pasted into the ABL Editor is indented for improved readability.                                      |
| Enable tabular formatting                       | Determines whether the Correct indentation command aligns. Tabular formatting helps you format code in columns. For example, the following shows code before tabular formatting:  
  DEFINE VARIABLE aChar1 AS CHARACTER NO-UNDO.  
  DEFINE VARIABLE aaChar22 AS CHARACTER NO-UNDO.  
  DEFINE VARIABLE aaaChar333 AS CHARACTER NO-UNDO.  
  After tabular formatting, the code is much easier to scan:  
  DEFINE VARIABLE aChar1 AS CHARACTER NO-UNDO.  
  DEFINE VARIABLE aaChar22 AS CHARACTER NO-UNDO.  
  DEFINE VARIABLE aaaChar333 AS CHARACTER NO-UNDO.  
  Note: Tabular formatting is enabled by default. If you do not want tabular formatting to be applied when you use the Correct Indentation command, turn the option off in Editor preferences. |
<p>| Enable &amp;OUT formatting                          | Determines whether &amp;OUT statements are indented in WebSpeed CGI Wrapper files.                                                               |
| Expand keywords                                 | Automatically expands abbreviated keywords in the current file to their fully spelled equivalents as you type.                            |</p>
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Case keywords</strong></td>
<td>Determines whether the ABL Editor immediately applies keyword casing when you type a space following a full or abbreviated keyword.</td>
</tr>
<tr>
<td><strong>No action</strong></td>
<td>Retains the original keyword casing for the existing ABL statements in the current file and applies uppercase to keywords in the newly added statements.</td>
</tr>
<tr>
<td><strong>Upper</strong></td>
<td>Applies uppercase to keywords in the existing ABL statements in the current file and to keywords in the newly added statements.</td>
</tr>
<tr>
<td><strong>Lower</strong></td>
<td>Applies lowercase to keywords in the existing ABL statements in the current file and to keywords in the newly added statements.</td>
</tr>
<tr>
<td><strong>Apply keyword casing on save</strong></td>
<td>Determines whether the ABL Editor applies keyword casing when you save the file.</td>
</tr>
<tr>
<td><strong>Expand database tables and fields</strong></td>
<td>Determines whether the ABL Editor expands a database table or field name as you type. For example, when selected, the Editor expands &quot;Cust&quot; to &quot;Customer&quot;.</td>
</tr>
<tr>
<td><strong>Automatically prefix fields and tables with database name</strong></td>
<td>Determines whether the ABL Editor qualifies database tables and fields with the database alias.</td>
</tr>
<tr>
<td><strong>Note</strong>: If multiple aliases are defined for a database connection, the first from the list is used.</td>
<td></td>
</tr>
<tr>
<td><strong>Enable smart home and end keys</strong></td>
<td>Determines where the Home and End keys reposition the cursor. When enabled, the Home key moves the cursor to the first non-white-space character of the current line. Pressing it again moves the cursor to the very beginning of the line. The End key moves the cursor to the first white-space character at the end of the current line. Pressing it again moves the cursor to the very last character of the line.</td>
</tr>
<tr>
<td><strong>Apply</strong></td>
<td>Saves your changes to the current workspace configuration.</td>
</tr>
<tr>
<td><strong>Restore Defaults</strong></td>
<td>Resets the preferences to the Progress Developer Studio for OpenEdge defaults.</td>
</tr>
</tbody>
</table>

See also
- Setting ABL Editor preferences on page 379
- ABL Editor Preferences on page 433

Editor Annotations options

The Annotations page (Progress OpenEdge > Editor > Annotations) of the Editor preferences lets you maintain a list of stored annotations that you can insert into ABL source files.
The page includes the following options:

<table>
<thead>
<tr>
<th>Annotation names</th>
<th>Lists all stored annotation templates. This list appears in the Adding annotations to source files on page 407 Add Annotation wizard on page 450. To browse through long lists, select a template and then drag the cursor up or down.</th>
</tr>
</thead>
</table>
| Annotation text  | Displays the text of the selected template. A template can include more than one annotation.  

**Note:** The templates that ship with the Annotation Generator contain two specially coded preprocessors, \%FILENAME\% and \%MODULE\%. \%FILENAME\% expands to the file name, including the extension. \%MODULE\% expands to the name of the directory where the file is stored. These are the only preprocessors the tool can expand. The tool does not support customer-created preprocessors. |
| Add              | Launches the **New Annotation** dialog.  

**Note:** The **New Annotation** dialog does not check the syntax of the template you create. |
| Edit             | Launches the **Edit Annotation** dialog for the selected annotation.  

**Note:** The **Edit Annotation** dialog does not check the syntax of the template you create. |
| Remove           | Removes the selected template from the internal template list. If you have exported your templates to a file, the template can be recovered by importing that file. |
| Import           | Overwrites the current annotation templates with the contents of an external .dat file. |
| Export           | Saves the current annotation templates to an external .dat file. |

Apply stores your changes to the current workspace configuration. Restore Defaults resets the preferences to the Progress Developer Studio for OpenEdge defaults.

**See also**
- Using annotations on page 406
- Setting ABL Editor preferences on page 379
- ABL Editor Preferences on page 433

**Editor Assistance options**

The Assistance page (Progress OpenEdge > Editor > Assistance) of the Editor preferences includes the following options:

| Show syntax help as I type | If checked, enables display of a pop-up box to prompt for remaining syntax above the line you are typing. This prompt is a brief reminder, not the full syntax shown with text hover. |
| **Highlight matching elements in code** | If checked, enables the display of a marker at the position of the paired code element (for example, a bracket or a keyword that begins or ends a code block) corresponding to the element at the current cursor position. |
| **Show description when proposing schema elements** | If checked, enables the inclusion of field descriptions in the right pane of the pop-up window when proposing completion options for schema references. See Balancing code-completion convenience and performance for more information. |
| **For schema** | If checked, enables display of a pop-up box identifying schema elements when you leave the cursor over them. |
| **For keyword** | If checked, enables display of a pop-up box containing a brief description and the syntax for ABL keywords and statements when you leave the cursor over them. |
| **Milliseconds allowed for re-parsing** | Specifies the maximum amount of time the ABL Editor will spend updating its analysis before building the list of completion proposals based on the last completed analysis. See Balancing code-completion convenience and performance for more information. |
| **Automatically propose completion on "," and ":"** | If checked, enables the auto-completion feature to activate whenever you type a period or colon in an appropriate place (in a class path for a period, in a database schema reference for a colon). If you disable this option, you can still access auto-completion by typing CTRL+SPACE. |
| **Auto-bracketing** | If checked, enables automatic insertion of the proper closing element (auto-bracketing) for any symbols that normally are used in pairs, such as parentheses, brackets, and quotation marks. |
| **Exclude proposals (list)** | Lets you selectively suppress specific types of code elements from assistance proposals. In the list, check the elements that you do not want to see. |
| **Order of proposals (list)** | Specifies the order in which assistance proposals are presented. To change the sequence, select an element and click Move Up or Move Down. |
| **Automatically add USING statement instead of qualified name** | If checked, adds the USING statement instead of qualified name. |

Apply stores your changes to the current workspace configuration. Restore Defaults resets the preferences to the Progress Developer Studio for OpenEdge defaults.

**See also**
- Using assistance features on page 389
- Setting ABL Editor preferences on page 379
- ABL Editor Preferences on page 433
## Editor Build options

The Build page (-progress openedge > Editor > Build) of the Editor preferences includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Compilable file extensions</strong></td>
<td>A comma-separated list of file extensions for file types that the OpenEdge Editor can compile. The default value is <code>p.w.cls.pgen</code>.</td>
</tr>
<tr>
<td><strong>AppBuilder file extensions</strong></td>
<td>A comma-separated list of file extensions for files to open in the AppBuilder view. The default value is <code>w</code>.</td>
</tr>
<tr>
<td><strong>Include file extensions</strong></td>
<td>A comma-separated list of file extensions for files that the OpenEdge Editor should treat as include files. The default value is <code>i</code>.</td>
</tr>
<tr>
<td><strong>SpeedScript extensions</strong></td>
<td>A comma-separated list of file extensions for files that the OpenEdge Editor should treat as SpeedScript files. The default value is <code>html</code>.</td>
</tr>
<tr>
<td><strong>Static files filter</strong></td>
<td>A comma-separated list of file extensions for static files that are not copied to the static files destination directory when the files change or a build runs. The default value is <code>project,propath,dbconnection</code>.</td>
</tr>
<tr>
<td><strong>Compile on save if required</strong></td>
<td>An option to have Progress Developer Studio for OpenEdge compile ABL source files each time you save them.</td>
</tr>
<tr>
<td></td>
<td>It is recommended that you select this option if you disable the Build automatically option in Eclipse Workspace preferences (preferences &gt; window &gt; general &gt; workspace). If automatic builds are enabled (the default setting), the <strong>Compile on save if required</strong> option has no effect.</td>
</tr>
<tr>
<td></td>
<td>You might choose to disable automatic builds if the contents of your workspace are frequently updated significantly (for example, by copying files from an external source), triggering build processes that block saving of resources until they finish. Otherwise, you should enable automatic builds to ensure that up-to-date r-code is always available for tools and features that require it.</td>
</tr>
<tr>
<td><strong>Automatically check syntax</strong></td>
<td>An option to have Progress Developer Studio for OpenEdge continuously check syntax and update related messages as you edit the ABL code.</td>
</tr>
<tr>
<td><strong>Show all syntax check message summaries</strong></td>
<td>An option to have the Check Syntax dialog display all available syntax-check messages, instead of only the first message. With this option enabled, each message includes a link that you can click to go to the applicable line in the source file.</td>
</tr>
</tbody>
</table>
Show help message detail with syntax check messages

An option to automatically include help text with each message listed in the Check Syntax dialog. Enabling this option has no effect unless Show all syntax check message summaries is also enabled.

Maximum number of errors reported per build

An integer specifying the number of build errors after which Progress Developer Studio for OpenEdge automatically terminates the build process. To have Progress Developer Studio for OpenEdge attempt to compile all files regardless of the number of errors, enter 0 or a negative integer in this field.

**Apply** stores your changes to the current workspace configuration. **Restore Defaults** resets the preferences to the Progress Developer Studio for OpenEdge defaults.

**See also**
- Compiling, testing, and running ABL programs on page 414
- Setting Editor options on page 379
- ABL Editor Preferences on page 433

**Editor Colors options**

You can use the **Colors** page to select the display colors for different code elements in the ABL Editor.

The **Colors** page of the Editor preferences includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| Comment    | The color for commented text. Select the **Bold** toggle to make all text in this category bold.  
Default setting - Green (63, 127, 95) |
| DataType   | The color for datatypes. Select the **Bold** toggle to make all text in this category bold.  
Default setting - Red (205, 58, 58) |
| Default    | The color for all text not covered by another category. Select the **Bold** toggle to make all text in this category bold.  
Default setting - Black (0,0,0) |
| Include    | The color for names of included files. Select the **Bold** toggle to make all text in this category bold.  
Default setting - Pink (255,0,128) |
| Keyword    | The color for ABL keywords. Select the **Bold** toggle to make all text in this category bold.  
Default setting - Purple (127,0,85) |
The color for hard-coded values, such as numbers and strings. Select the **Bold** toggle to make all text in this category bold.  
Default setting - Blue (42,0,255)

The color for preprocessors. Select the **Bold** toggle to make all text in this category bold.  
Default setting - Brown (139,126,102)

The color for inactive preprocessor regions. Select the **Bold** toggle to make all text in this category bold.  
Default setting - Gray (179,166,142)

The color for schema elements, such as database tables and fields. Select the **Bold** toggle to make all text in this category bold.  
Default setting - Orange (255,128,0)

The color for names of SpeedScript files. Select the **Bold** toggle to make all text in this category bold.  
Default setting - Black (0,0,0)

The syntax coloring page for HTML, CSS, and JavaScript.

Resets the preferences to the Progress Developer Studio for OpenEdge defaults.

Saves your changes to the current workspace configuration.

## See also
- Setting ABL Editor preferences on page 379
- Selecting a display color for code element on page 398
- ABL Editor Preferences on page 433

## Editor Templates (Macros) options

The **Templates** page lets you manage the set of macros, or templates, available in your workspace. This page appears when you select **Window > Preferences > Progress OpenEdge > Editor > Templates (Macros)**.

This page provides the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template list</td>
<td>Displays the available templates. Templates that are checked are available for use; those that are not checked are disabled.</td>
</tr>
<tr>
<td>Preview</td>
<td>Displays the pattern for the selected template.</td>
</tr>
<tr>
<td>New</td>
<td>Launches the New Template dialog to add a template.</td>
</tr>
</tbody>
</table>
Launches the Edit Template dialog to enable you to edit the definition of the selected template.

Deletes the selected templates.

Undoes any deletions.

Undoes changes to the selected templates, resetting them to their default definitions. This command is available only when all selected entries are predefined templates installed with Progress Developer Studio for OpenEdge.

Imports an XML file of template definitions.

Writes the selected template definitions to an XMI file, abbreviations.xml by default.

Resets the list of templates to its as-installed state, undoing all modifications.

Stores your changes to the current workspace configuration.

**Note:** Macros created in versions earlier than OpenEdge 10.1C are not compatible with the current version and must be migrated to templates before you can use them. If you have such macros in your workspace, select **OpenEdge > Migration > Migrate 10.1A or 10.1B Macros to Templates** to launch the migration tool.

**Default ABL Editor templates**

Default ABL Editor templates are listed in the table below. The string ${cursor} in the pattern is a control sequence that is not expanded, but rather determines where the cursor is positioned after the template is expanded.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Description</th>
<th>Associated Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;FR	</td>
<td>Frame name	</td>
<td>{&amp;FRAME-NAME} ${cursor}</td>
</tr>
<tr>
<td>&amp;GL</td>
<td>Global name</td>
<td>&amp;GLOBAL-DEFINE ${cursor}</td>
</tr>
<tr>
<td>&amp;SC</td>
<td>Scope defined</td>
<td>&amp;SCOPED-DEFINE ${cursor}</td>
</tr>
<tr>
<td>CMT</td>
<td>Comment</td>
<td>/* ${cursor} */</td>
</tr>
<tr>
<td>DED</td>
<td>Event with delegate</td>
<td>DEFINE EVENT ${cursor} DELEGATE</td>
</tr>
<tr>
<td>DES</td>
<td>Event with signature</td>
<td>DEFINE EVENT ${cursor} SIGNATURE</td>
</tr>
<tr>
<td>DVCH</td>
<td>Character variable</td>
<td>DEFINE VARIABLE ${cursor} AS CHARACTER</td>
</tr>
<tr>
<td>DVDE</td>
<td>Decimal variable</td>
<td>DEFINE VARIABLE ${cursor} AS DECIMAL</td>
</tr>
<tr>
<td>Sequence</td>
<td>Description</td>
<td>Associated Pattern</td>
</tr>
<tr>
<td>----------</td>
<td>----------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td>DVDT</td>
<td>Date variable</td>
<td>DEFINE VARIABLE ${cursor} AS DATE</td>
</tr>
<tr>
<td>DVHN</td>
<td>Handle variable</td>
<td>DEFINE VARIABLE ${cursor} AS HANDLE</td>
</tr>
<tr>
<td>DVIN</td>
<td>Integer variable</td>
<td>DEFINE VARIABLE ${cursor} AS INTEGER</td>
</tr>
<tr>
<td>DVLG</td>
<td>Logical variable</td>
<td>DEFINE VARIABLE ${cursor} AS LOGICAL</td>
</tr>
<tr>
<td>DVMP</td>
<td>CHAR variable</td>
<td>DEFINE VARIABLE ${cursor} AS MEMPTR</td>
</tr>
<tr>
<td>DVRI</td>
<td>ROWID variable</td>
<td>DEFINE VARIABLE ${cursor} AS ROWID</td>
</tr>
<tr>
<td>DVWH</td>
<td>Widget variable</td>
<td>DEFINE VARIABLE ${cursor} AS WIDGET</td>
</tr>
<tr>
<td>IOPCH</td>
<td>Character INPUT-OUTPUT</td>
<td>DEFINE INPUT-OUTPUT PARAMETER ${cursor} AS CHARACTER</td>
</tr>
<tr>
<td>IOPDE</td>
<td>Decimal INPUT-OUTPUT</td>
<td>DEFINE INPUT-OUTPUT PARAMETER ${cursor} AS DECIMAL</td>
</tr>
<tr>
<td>IOPDT</td>
<td>Date INPUT-OUTPUT</td>
<td>DEFINE INPUT-OUTPUT PARAMETER ${cursor} AS DATE</td>
</tr>
<tr>
<td>IOPHN</td>
<td>Handle INPUT-OUTPUT</td>
<td>DEFINE INPUT-OUTPUT PARAMETER ${cursor} AS HANDLE</td>
</tr>
<tr>
<td>IOPIN</td>
<td>Integer INPUT-OUTPUT</td>
<td>DEFINE INPUT-OUTPUT PARAMETER ${cursor} AS INTEGER</td>
</tr>
<tr>
<td>IOPLG</td>
<td>Logical INPUT-OUTPUT</td>
<td>DEFINE INPUT-OUTPUT PARAMETER ${cursor} AS LOGICAL</td>
</tr>
<tr>
<td>IOPMP</td>
<td>Memptr INPUT-OUTPUT</td>
<td>DEFINE INPUT-OUTPUT PARAMETER ${cursor} AS MEMPTR</td>
</tr>
<tr>
<td>IOPRI</td>
<td>ROWID INPUT-OUTPUT</td>
<td>DEFINE INPUT-OUTPUT PARAMETER ${cursor} AS ROWID</td>
</tr>
<tr>
<td>IOPWH</td>
<td>Widget INPUT-OUTPUT</td>
<td>DEFINE INPUT-OUTPUT PARAMETER ${cursor} AS WIDGET</td>
</tr>
<tr>
<td>IPCH</td>
<td>Character INPUT</td>
<td>DEFINE INPUT PARAMETER ${cursor} AS CHARACTER</td>
</tr>
<tr>
<td>IPDE</td>
<td>Decimal INPUT</td>
<td>DEFINE INPUT PARAMETER ${cursor} AS DECIMAL</td>
</tr>
<tr>
<td>IPDT</td>
<td>Date INPUT</td>
<td>DEFINE INPUT PARAMETER ${cursor} AS DATE</td>
</tr>
<tr>
<td>IPHN</td>
<td>Handle INPUT</td>
<td>DEFINE INPUT PARAMETER ${cursor} AS HANDLE</td>
</tr>
</tbody>
</table>
### Sequence Description Associated Pattern

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Description</th>
<th>Associated Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPIN</td>
<td>Integer INPUT</td>
<td>DEFINE INPUT PARAMETER ${cursor} AS INTEGER</td>
</tr>
<tr>
<td>IPLG</td>
<td>Logical INPUT</td>
<td>DEFINE INPUT PARAMETER ${cursor} AS LOGICAL</td>
</tr>
<tr>
<td>IPMP</td>
<td>Memptr INPUT</td>
<td>DEFINE INPUT PARAMETER ${cursor} AS MEMPTR</td>
</tr>
<tr>
<td>IPRI</td>
<td>ROWID INPUT</td>
<td>DEFINE INPUT PARAMETER ${cursor} AS ROWID</td>
</tr>
<tr>
<td>IPWH</td>
<td>Widget INPUT</td>
<td>DEFINE INPUT PARAMETER ${cursor} AS WIDGET</td>
</tr>
<tr>
<td>MES</td>
<td>Message</td>
<td>MESSAGE ${cursor}</td>
</tr>
<tr>
<td>OPCH</td>
<td>Char OUTPUT</td>
<td>DEFINE OUTPUT PARAMETER ${cursor} AS CHARACTER</td>
</tr>
<tr>
<td>OPDE</td>
<td>Decimal OUTPUT</td>
<td>DEFINE OUTPUT PARAMETER ${cursor} AS DECIMAL</td>
</tr>
<tr>
<td>OPDT</td>
<td>Date OUTPUT</td>
<td>DEFINE OUTPUT PARAMETER ${cursor} AS DATE</td>
</tr>
<tr>
<td>OPHN</td>
<td>Handle OUTPUT</td>
<td>DEFINE OUTPUT PARAMETER ${cursor} AS HANDLE</td>
</tr>
<tr>
<td>OPIN</td>
<td>Integer OUTPUT</td>
<td>DEFINE OUTPUT PARAMETER ${cursor} AS INTEGER</td>
</tr>
<tr>
<td>OPLG</td>
<td>Logical OUTPUT</td>
<td>DEFINE OUTPUT PARAMETER ${cursor} AS LOGICAL</td>
</tr>
<tr>
<td>OPMP</td>
<td>Memptr OUTPUT</td>
<td>DEFINE OUTPUT PARAMETER ${cursor} AS MEMPTR</td>
</tr>
<tr>
<td>OPRI</td>
<td>ROWID OUTPUT</td>
<td>DEFINE OUTPUT PARAMETER ${cursor} AS ROWID</td>
</tr>
<tr>
<td>OPWH</td>
<td>Widget OUTPUT</td>
<td>DEFINE OUTPUT PARAMETER ${cursor} AS WIDGET</td>
</tr>
</tbody>
</table>

**See also**

- Using templates for easier text entry on page 382
- Setting ABL Editor preferences on page 379
- ABL Editor Preferences on page 433

### Class Cache options

The **Class Cache** page (Window > Preferences > Progress OpenEdge > Advanced > Class Cache) includes the following options:

**Class cache**
## Parsing options

The Parsing preferences page provide options which allow you to optimally build the Abstract Syntax Tree (AST) for an include file.

Some of the programs might have many references of the same include file. Such programs cause the Progress Developer Studio for OpenEdge to consume more memory, because each include file has a separate AST. The Includes file preferences section on the Parsing preferences page allows you manage the include file references of a program while constructing an AST.

You can access the Parsing preferences page by selecting Window > Preferences > Progress OpenEdge > Advanced > Parsing.

The Parsing preferences page includes the following options:

| **Limit scope of cache class information to** | Define what directories are searched for classes and interfaces when building the cache. With the Workspace option, Progress Developer Studio for OpenEdge caches information only for classes that are in the current workspace. Using the PROPATH option expands the class cache scope and provides more functionality when working with referenced classes and interfaces that are outside the workspace. However, the PROPATH option can make startup of Progress Developer Studio for OpenEdge slower, as building the class cache takes longer. |
| **Scan installation directory at startup for procedure information** | Select to search the directory in which OpenEdge is installed for procedures and graphical interface (.w) files. If this option is not selected, this directory and its subdirectories are not searched, even if the class cache scope is set to PROPATH. |

| **Cache refresh** |  |
| **OpenEdge Database** | Select to refresh schemas for connected OpenEdge databases whenever **OpenEdge > Refresh cache** is selected from the main menu bar of Progress Developer Studio for OpenEdge. |
| **OpenEdge Super Procedures** | Select to refresh OpenEdge super procedures whenever **OpenEdge > Refresh cache** is selected from the main menu bar of Progress Developer Studio for OpenEdge. |

**Apply** stores your changes to the current workspace configuration. **Restore Defaults** resets the preferences to the Progress Developer Studio for OpenEdge defaults.

### See also

- [Setting ABL Editor preferences on page 379](#)
- [ABL Editor Preferences on page 433](#)
Cache the file content

Select this option to cache the include file contents. This stores the contents of an include file in the memory until the program is closed.

Trim the content

Select this option to trim an include file's content. Only the definitions used in the Outline view and the entities (tokens) required to understand the structure of an include file are considered while constructing an AST, and other information of an include file is discarded.

For example, only the procedure declaration and the parameters of an internal procedure present in an include file are considered while constructing an AST. The other statements of an internal procedure are discarded.

Note: This option is enabled when you select the Cache the file content check box.

See also

page 379

USING Statements options

You can use the USING Statements preferences page to control the USING statements' functionality in your current workspace.

This page provides the following controls:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display USING statements</td>
<td>Lets you choose whether you want to display USING statements:</td>
</tr>
<tr>
<td></td>
<td>• After the error-handling statement or,</td>
</tr>
<tr>
<td></td>
<td>• Before the error-handling statement.</td>
</tr>
<tr>
<td>Group all USING statements together</td>
<td>Specifies whether to relocate the existing statements to a new location or not. The relocation is based on the Display USING statements preference.</td>
</tr>
<tr>
<td></td>
<td>If the Group all USING statements together option is selected, then all the USING statements (existing and new) are placed before or after the error-handling statement based on the selection.</td>
</tr>
<tr>
<td></td>
<td>If this option is not selected and there are existing USING statements, then the new USING statements are appended to the last USING statement.</td>
</tr>
<tr>
<td>Sort statements alphabetically</td>
<td>Sorts all the grouped USING statements in alphabetical order.</td>
</tr>
<tr>
<td></td>
<td>Note: This option is enabled only when the Group all USING statements together option is selected.</td>
</tr>
</tbody>
</table>
Specifies whether to use individual type names or wildcards as part of the USING declaration.

If you select the **Qualified type names** option, then all the existing and new USING statements are expanded to their fully qualified type.

If you select the **Wildcards (.* )** option, then all the existing and new USING statements use wildcards in the USING declaration.

**Note:** For the changes to occur, the **Preserve valid USING statements** option must not be selected.

Specifies whether or not to add the FROM ASSEMBLY/FROM PROPATH option to USING statements.

If this option is selected, all the existing and newly added USING statements contain the FROM ASSEMBLY/FROM PROPATH option.

If this option is not selected, all the existing USING statements retain the FROM ASSEMBLY/FROM PROPATH option (if it is already added), but the newly added USING statements do not include FROM ASSEMBLY/FROM PROPATH option.

Preserves the existing, valid USING statements in their original form and displays the newly added statements based on whether the style preference is selected as **Qualified type names** or **Wildcards (.* )**.

**Note:** To prevent existing, valid statements from being relocated, you must not select the **Group all USING statements together** option and you must select the **Preserve valid USING statements**.

### ABL Editor menus

#### Source menu

The **Source** menu contains options specific to the ABL Editor. The **Source** menu is available from the **Source** main menu and from the editor's context (right-click) menu.

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify USING statement style</td>
<td>Specifies whether to use individual type names or wildcards as part of the USING declaration. If you select the <strong>Qualified type names</strong> option, then all the existing and new USING statements are expanded to their fully qualified type. If you select the <em><em>Wildcards (.</em> )</em>* option, then all the existing and new USING statements use wildcards in the USING declaration. <strong>Note:</strong> For the changes to occur, the <strong>Preserve valid USING statements</strong> option must not be selected.</td>
</tr>
<tr>
<td>Include FROM ASSEMBLY/FROM PROPATH option</td>
<td>Specifies whether or not to add the FROM ASSEMBLY/FROM PROPATH option to USING statements. If this option is selected, all the existing and newly added USING statements contain the FROM ASSEMBLY/FROM PROPATH option. If this option is not selected, all the existing USING statements retain the FROM ASSEMBLY/FROM PROPATH option (if it is already added), but the newly added USING statements do not include FROM ASSEMBLY/FROM PROPATH option.</td>
</tr>
<tr>
<td>Preserve valid USING statements</td>
<td>Preserves the existing, valid USING statements in their original form and displays the newly added statements based on whether the style preference is selected as <strong>Qualified type names</strong> or *<em>Wildcards (.</em> )**. <strong>Note:</strong> To prevent existing, valid statements from being relocated, you must not select the <strong>Group all USING statements together</strong> option and you must select the <strong>Preserve valid USING statements</strong>.</td>
</tr>
</tbody>
</table>

The Check Syntax option checks the ABL syntax of the current file. This option does not save an r-code version of the file. If you select **Correct Case**, it applies the user preference for keyword casing for keywords in the current file. **Expand Keywords** replaces abbreviated keywords in the current file with their fully spelled equivalents.
<table>
<thead>
<tr>
<th>Correct Indentation</th>
<th>Indents lines in the current file to improve readability.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organize USING Statements</td>
<td>Organizes USING statements in ABL sources that are currently open or selected. It removes duplicate and unused USING statements and sorts the required ones in alphabetical order. It expands . * to a fully qualified path. It also prompts you if a referenced type name cannot be mapped uniquely to a type in the current project.</td>
</tr>
<tr>
<td>Update Function Prototypes</td>
<td>Modifies the signatures of function prototypes in the current file to match the signatures of the corresponding function declarations, as necessary.</td>
</tr>
<tr>
<td>Strip AppBuilder markup</td>
<td>Removes the AppBuilder markup code from a file originally created in AppBuilder or by means of the New ABL Structured Include or the New ABL Structured Procedure wizard.</td>
</tr>
<tr>
<td>Caution:</td>
<td>If this code is removed, the file cannot be opened in AppBuilder again.</td>
</tr>
<tr>
<td>Add Constructor</td>
<td>Opens the Add Constructor wizard to add a constructor to the current class.</td>
</tr>
<tr>
<td>Add Static Constructor</td>
<td>Adds a static constructor to the current class.</td>
</tr>
<tr>
<td>Add Constructors from Super Class</td>
<td>Opens the Add Constructors from Super Class wizard to select any constructors with arguments from the immediate super class to override in the current class.</td>
</tr>
<tr>
<td>Add Destructor</td>
<td>Adds a destructor to the current class.</td>
</tr>
<tr>
<td>Add Method</td>
<td>Opens the Add Method wizard to add a method to the current class.</td>
</tr>
<tr>
<td>Add Event</td>
<td>Opens the Add Event wizard to add an event to the current class.</td>
</tr>
<tr>
<td>Add Property</td>
<td>Opens the Add Property wizard to add a property to the current class.</td>
</tr>
<tr>
<td>Override/Implement Members</td>
<td>Opens the Override/Implement Members wizard to select members from the class hierarchy to override or implement in the current class.</td>
</tr>
<tr>
<td>Surround With</td>
<td>Shows a submenu that lets you surround the selected code segment with a CATCH block or a FINALLY block for structured error handling.</td>
</tr>
<tr>
<td>Add Procedure</td>
<td>Opens the Add Procedure wizard to add an internal procedure to the current procedure.</td>
</tr>
<tr>
<td>Add Function</td>
<td>Opens the Add Function wizard on page 453 to add a function to the current procedure.</td>
</tr>
</tbody>
</table>
Add Annotation | Opens the Add Annotation wizard on page 450 to insert ABL annotations in one or more files.

Define Service Interface | Opens the Define Service Interface wizard that allows you to publish the selected procedures and functions as a service interface.

Note: This option is available only in the view source mode.

See also
Writing ABL code on page 381
Editor context menu on page 448

ABL Editor context menu

Right-clicking in a file displays the ABL Editor context menu. This menu has the following entries:

<table>
<thead>
<tr>
<th>Entry</th>
<th>DescriptionAZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undo</td>
<td>Undo the last action in the current file.</td>
</tr>
<tr>
<td>Revert file</td>
<td>Replace the current file with the local saved copy.</td>
</tr>
<tr>
<td>Save</td>
<td>Save the current file.</td>
</tr>
<tr>
<td>View Design (Windows only)</td>
<td>Show the form in the Visual Designer. Although this command is enabled for all class files, an error results if the file is not a valid OpenEdge GUI for .NET class.</td>
</tr>
<tr>
<td>Show In</td>
<td>See the Workbench User Guide.</td>
</tr>
<tr>
<td>Quick Outline</td>
<td>Open the Quick Outline window.</td>
</tr>
<tr>
<td>Cut</td>
<td>Cut the selected text.</td>
</tr>
<tr>
<td>Copy</td>
<td>Copy the selected text.</td>
</tr>
<tr>
<td>Paste</td>
<td>Paste the contents of the Clipboard.</td>
</tr>
<tr>
<td>Shift Right</td>
<td>Increase the indent of the current line or the highlighted lines.</td>
</tr>
<tr>
<td>Shift Left</td>
<td>Decrease the indent of the current line or the highlighted lines.</td>
</tr>
<tr>
<td>Source</td>
<td>Display the Progress Developer Studio for OpenEdge Source menu.</td>
</tr>
<tr>
<td>Check Syntax</td>
<td>Check the ABL syntax of the current file. This does not save an r-code version of the file.</td>
</tr>
<tr>
<td>Compile</td>
<td>Compile the current file and save the r-code.</td>
</tr>
<tr>
<td>Keyword Help</td>
<td>Open the Keyword Help view in the Console area, displaying the OpenEdge Development: ABL Reference entry for the selected keyword.</td>
</tr>
<tr>
<td>Toggle Breakpoint</td>
<td>Set or remove a breakpoint at the current line for the OpenEdge Debugger.</td>
</tr>
<tr>
<td>---------------------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Show References</td>
<td>Open the References dialog which displays the external references in the current file.</td>
</tr>
<tr>
<td>Open Declaration</td>
<td>Move the cursor to the beginning of the code that defines the variable, function, procedure, or method whose name you are pointing to when you issue the command. The declaration can be in the current file or an external file; if necessary. OpenEdge opens the external file.</td>
</tr>
<tr>
<td>Run As</td>
<td>Display the OpenEdge Run menu.</td>
</tr>
<tr>
<td>Debug As</td>
<td>Display a submenu of appropriate debuggers for the current file.</td>
</tr>
<tr>
<td>Team</td>
<td>See the Workbench User Guide.</td>
</tr>
<tr>
<td>Compare With</td>
<td>See the Workbench User Guide.</td>
</tr>
<tr>
<td>Replace With</td>
<td>See the Workbench User Guide.</td>
</tr>
<tr>
<td>Progress OpenEdge</td>
<td>Display a sub-menu of common Progress OpenEdge actions.</td>
</tr>
<tr>
<td>Preferences</td>
<td>Launch the Preferences window displaying the General &gt; Editors &gt; Text Editors preference page. See the Workbench User Guide for more information on these settings.</td>
</tr>
</tbody>
</table>

**See also**

Writing ABL code on page 381
Source menu on page 446

**Progress OpenEdge context menu**

The Progress OpenEdge menu contains options specific to the ABL Editor and is available from the editor's right-click context menu.

<table>
<thead>
<tr>
<th>Define Service Interface</th>
<th>Check the ABL syntax of the current file. This option does not save an r-code version of the file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exclude from Build</td>
<td>Enables you to exclude compilable resources when you build your project. You can exclude files, folders, or both from the build.</td>
</tr>
<tr>
<td>Include in Build</td>
<td>Enables you to include the excluded resources to the build. This option is only available for the resources that were excluded from the build.</td>
</tr>
<tr>
<td>Add Annotations</td>
<td>Open the Add Annotation wizard to insert ABL annotations in one or more files.</td>
</tr>
<tr>
<td>Restart OpenEdge AVM</td>
<td>Enables you to restart the OpenEdge AVM (ABL Virtual Machine).</td>
</tr>
<tr>
<td>Compile</td>
<td>Enables you to compile the selected project files or folders.</td>
</tr>
</tbody>
</table>
Clear OpenEdge Compile Errors

Clear any OpenEdge error for a particular file or project folder from the Problems view.

Note: The Clear OpenEdge Compile Errors option does not correct the errors. Any unfixed errors are flagged again when you next compile the code.

Generate BPM Invocation Files

Enables you to generate a .bizoe file for the OpenEdge procedure file. The .bizoe file is required for the procedure file to communicate with the BPM process.

Generate Sonic ESB Invocation Files

Enables you to generate Sonic .esboe file for the OpenEdge procedure file.

Generate REST OE Invocation Files

Enables you to generate REST .pidl file for the OpenEdge procedure file.

See also
Writing ABL code on page 381
Editor context menu on page 448

Code-generation wizards

This section provides detailed information about the various wizards available in the ABL Editor for creating ABL source files and code segments.

Add Annotation wizard

The Add Annotation wizard lets you add a new or existing ABL annotation to a selected set of ABL procedure and include files in open projects in your workspace.

This wizard appears when you select Source > Add Annotation> from the main menu bar or the ABL Editor context (right-click) menu, or when you select Progress OpenEdge > Add Annotation from the context menu for the Project Explorer view or the ABL Editor. The following controls are available:

Annotation selector
Displays a drop-down list of currently defined annotations available for use. You can manage the contents of this list at the Annotations preferences page.

Annotation template editor
Displays and lets you edit the content of the annotation to be inserted. The Edit button opens a dialog that supports changes specific to ESB annotations; this button is not active for other annotation types.

Available resources filter
Lets you filter the list of resources displayed in the treeview below.
On the first page of the wizard - Let you select the files to which the annotation is to be added. If you finish the wizard without making any entries on the second page, the annotation is inserted at the beginning of each selected file.

Note: If the selected annotation is ESB - Main, class files do not appear in the resources list because the OpenEdge Adapter for Sonic ESB does not support class files.

On the second page of the wizard - Let you select specific points (functions, procedures, and data members within the selected files) where the annotation is to be added.

Show selected
If checked, filters the list of available resources to show only those that you have selected.

Back up selected resources
If checked, enables the Location field to let you specify a directory where copies of the original (un-annotated) selected resources are stored.

Location
Optionally specifies a directory for backup copies of the selected resources.

See also
Using annotations on page 406
Editor Annotations options on page 435

Add Constructor wizard
The Add Method wizard helps you add a constructor method to a class file.
This wizard appears when you select Source > Add Constructor.

The following controls are available:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public/Protected/Private</td>
<td>Specifies the access modifier that applies to the method:</td>
</tr>
<tr>
<td></td>
<td>• Public - The constructor can be called from any class and directly</td>
</tr>
<tr>
<td></td>
<td>invoked in any derived class.</td>
</tr>
<tr>
<td></td>
<td>• Protected - The constructor can be directly invoked in any derived class,</td>
</tr>
<tr>
<td></td>
<td>but it cannot be called from a class that is not derived from the</td>
</tr>
<tr>
<td></td>
<td>declaring class.</td>
</tr>
<tr>
<td></td>
<td>• Private - The constructor cannot be called from any other class and</td>
</tr>
<tr>
<td></td>
<td>cannot be directly invoked in a derived class.</td>
</tr>
<tr>
<td>Insert catch block</td>
<td>If checked, adds a CATCH block to the generated method stub to support</td>
</tr>
<tr>
<td></td>
<td>structured error handling.</td>
</tr>
</tbody>
</table>
If checked, adds a FINALLY block to the generated method stub to support structured error handling.

Determines where the constructor code is inserted in the source file.

See also
Adding constructors on page 385
Adding constructors from a super class on page 452
Adding a destructor on page 386
Writing object-oriented code on page 385

Add Constructors from Super Class wizard

The Override/Implement Members wizard helps you to:

- Locate inherited data members that you want to override.
- Satisfy the requirement to implement data members declared in interfaces implemented by the current class, and to implement inherited abstract members.

This wizard appears when you select Source > Override/Implement Members. The following controls are available:

<table>
<thead>
<tr>
<th>Constructor list</th>
<th>Displays and lets you select constructor methods declared in the parent class from which the current class inherits.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insertion position</td>
<td>Determines where the constructor code is inserted in the source file.</td>
</tr>
<tr>
<td>Generate method comments</td>
<td>If checked, adds a comment block before the inserted method code.</td>
</tr>
</tbody>
</table>

See also
Adding constructors on page 385
Adding constructors from a super class on page 452
Adding a destructor on page 386
Writing object-oriented code on page 385

Add Event wizard

The Add Event wizard helps you add event code to a class file.

This wizard appears when you select Source > Add Event.

The following controls are available:

| Event name | (Required) Specifies the name of the event. The name must begin with a letter; spaces and most non-alphanumeric characters are not allowed. |
**Public/Protected/Private**

Specifies the access modifier that applies to the event:

- **Public** - The event can be called from any class and can be directly invoked in any derived class.
- **Protected** - The event can be directly invoked in any derived class, but it cannot be called from a class that is not derived from the declaring class.
- **Private** - The event cannot be called from any other class and cannot be directly invoked in a derived class. A private event cannot be abstract.

**Abstract**

(Enabled only if current class is abstract) If checked, specifies that the event is abstract and cannot be instantiated. An abstract event is designed to be inherited and to be implemented by the inheriting class. Therefore, an abstract event cannot be private, nor can it be static.

**Static**

If checked, specifies that the event is static. A static event is accessible by every instance of the class and (if public) by clients of the class without the need to create an instance of the class. A static event cannot be abstract.

**Delegate**

If checked, enables the field to the right in which you can specify a class that serves as the event delegate.

**Insertion position**

Determines where the event code is inserted in the source file.

**See also**

Adding events on page 387
Writing object-oriented code on page 385

**Add Function wizard**

The Add Procedure wizard helps you add a function to an ABL procedure or include file.

This wizard appears when you select **Source > Add Function**.

The following controls are available:

**Function**

(Required) Select this option to create a function. Enter a function name in the **Name** field.

**Note:** This option is available only if the current file open in the ABL Editor is an ADM (adm1 or adm2) file.
Override
(Required) Select this option to have the Name field display a list of the functions for which you can create overrides. This list is the user-defined functions of all Super Procedures defined for the procedure file (excluding those user-defined functions defined as PRIVATE). This option is disabled when there are no functions to override.

Note: This option is available only if the current file open in the ABL Editor is an ADM (adm1 or adm2) file.

Name
(Required) Specifies the name of the function.

Insert catch block
If checked, adds a CATCH block to the generated procedure stub to support structured error handling.

Return type
Specifies the data type of the value returned by the function.

Extent
If checked, specifies that the value returned by the function is an extent (array) value and enables the adjacent field to the right where you enter the number of elements in the extent.

Insert catch block
If checked, adds a CATCH block to the generated function stub to support structured error handling.

Insert finally block
If checked, adds a FINALLY block to the generated function stub to support structured error handling.

Private
Specifies the access modifier that applies to a function. If selected, the function cannot called from any other ABL procedure file.

DB-Required
If checked, a preprocessor directive is wrapped around the generated function allowing it to be conditionally excluded from compilation. For more information, see Specifying DB-Required code blocks.

Note: A function that is not DB-Required is excluded from a data object's client proxy r-code file and only included in the data object's main r-code file.

Insertion position
Determines where the function code is inserted in the source file.

See also
Adding functions on page 383
Writing procedural code on page 384

Add Method wizard
The Add Method wizard helps you add method code to a class file. This wizard appears when you select Source > Add Method.
The following controls are available:

<table>
<thead>
<tr>
<th>Method name</th>
<th>(Required) Specifies the name of the method. The name must begin with a letter; spaces and most non-alphanumeric characters are not allowed.</th>
</tr>
</thead>
</table>
| Public/Protected/Private | Specifies the access modifier that applies to the method:  
• **Public** - The method can be called from any class and can be directly invoked in any derived class.  
• **Protected** - The method can be directly invoked in any derived class, but it cannot be called from a class that is not derived from the declaring class.  
• **Private** - The method cannot be called from any other class and cannot be directly invoked in a derived class. A private method cannot be abstract. |
| Abstract | (Enabled only if current class is abstract) If checked, specifies that the method is abstract and cannot be instantiated. An abstract event is designed to be inherited and to be implemented by the inheriting class. Therefore, an abstract event cannot be private, nor can it be static. |
| Static | If checked, specifies that the method is static. A static method is accessible by every instance of the class and (if public) by clients of the class without the need to create an instance of the class.  
A static method cannot be abstract. |
| Return type | Specifies the method's data type. |
| Insert catch block | If checked, adds a CATCH block to the generated method stub to support structured error handling. |
| Insert finally block | If checked, adds a FINALLY block to the generated method stub to support structured error handling. |
| Insertion position | Determines where the method code is inserted in the source file. |

**See also**

Adding methods on page 386  
Writing object-oriented code on page 385

**Add Procedure wizard**

The **Add Procedure** wizard helps you add internal procedure code to an ABL procedure or include file.  
This wizard appears when you select **Source > Add Procedure**.  
The following controls are available:
### See also

- Adding internal procedures on page 384
- Writing procedural code on page 383

### Add Property wizard

The Add Property wizard helps you add property code to a class file.

This wizard appears when you select **Source > Add Property**.

The following controls are available:

<table>
<thead>
<tr>
<th><strong>Procedure</strong></th>
<th>(Required) Select this option to create a procedure. Enter a procedure name in the <strong>Procedure name</strong> field.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>This option is available only if the current file open in the ABL Editor is an ADM (adm1 or adm2) file.</td>
</tr>
<tr>
<td><strong>Override</strong></td>
<td>(Required) Select this option to have the <strong>Procedure name</strong> field display a list of the procedures for which you can create overrides. This list is the internal procedures of all or Super Procedures defined for the procedure file (excluding those internal procedures defined as PRIVATE). This option is disabled when there are no procedures to override.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>This option is available if the current file open in the ABL Editor is an ADM (adm1 or adm2) file.</td>
</tr>
<tr>
<td><strong>Procedure name</strong></td>
<td>(Required) Specifies the name of the procedure.</td>
</tr>
<tr>
<td><strong>Insert catch block</strong></td>
<td>If checked, adds a CATCH block to the generated procedure stub to support structured error handling.</td>
</tr>
<tr>
<td><strong>Insert finally block</strong></td>
<td>If checked, adds a FINALLY block to the generated procedure stub to support structured error handling.</td>
</tr>
<tr>
<td><strong>Private</strong></td>
<td>Specifies the access modifier that applies to a procedure. If selected, the procedure cannot called from any other ABL procedure file.</td>
</tr>
<tr>
<td><strong>DB-Required</strong></td>
<td>If checked, a preprocessor directive is wrapped around the generated procedure allowing it to be conditionally excluded from compilation. For more information, see Specifying DB-Required code blocks.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>A procedure that is not DB-Required is excluded from a data object's client proxy r-code file and only included in the data object's main r-code file.</td>
</tr>
<tr>
<td><strong>Insertion position</strong></td>
<td>Determines where the procedure code is inserted in the source file.</td>
</tr>
<tr>
<td>Property name</td>
<td>(Required) Specifies the name of the property. The name must begin with a letter; spaces and most non-alphanumeric characters are not allowed.</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Public/Protected/Private | Specifies the access modifier that applies to the property, GET accessor, or SET accessor:  
  • **Public** - The property or accessor can be called from any class and can be directly invoked in any derived class.  
  • **Protected** - The property or accessor can be directly invoked in any derived class, but it cannot be called from a class that is not derived from the declaring class.  
  • **Private** - The property or accessor cannot be called from any other class and cannot be directly invoked in a derived class. A private property cannot be abstract.  
  The access modifier for an accessor cannot be less restrictive than the modifier for the property itself. |
| Static | If checked, specifies that the property is static. A static property is accessible by every instance of the class and (if public) by clients of the class without the need to create an instance of the class.  
A static property cannot be abstract. |
| Abstract | (Enabled only if the current class is abstract) If checked, specifies that the property is abstract and cannot be instantiated. An abstract property is designed to be inherited and to be implemented by the inheriting class. Therefore, an abstract property cannot be private, nor can it be static. |
| Type | Specifies the property's data type. |
| Extent | If checked, specifies that the property is an extent (array) value and enables the adjacent field to the right where you enter the number of elements in the extent. |
| Initial value | If checked, enables the adjacent field to the right where you enter the initial value of the property. |
| NO-UNDO | If checked, adds the NO-UNDO attribute to the property declaration. |
| Readable | If checked, specifies that the GET accessor is readable. |
| Writable | If checked, specifies that the SET accessor is writable. |
| Insert implementation | If checked, adds an implementation stub for the accessor to the generated code. |
| Insertion position | Determines where the property code is inserted in the source file. |

**See also**

*Adding properties* on page 387
New ABL Class wizard

The **New ABL Class** wizard helps you create a class file.

This wizard appears when you select **File > New > ABL Class**.

The following controls are available:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package root</td>
<td>Specifies a currently open project to contain the class code and other project code. Click <strong>Browse</strong> if you want to select a project other than the current one (the default value).</td>
</tr>
<tr>
<td>Package</td>
<td>Optionally specifies a package name, corresponding to a sub-folder of the package root, to contain the class file. Click <strong>Browse</strong> or enter the folder path by typing. A period (.) must separate each subfolder from its parent folder in the path name.</td>
</tr>
<tr>
<td>Class name</td>
<td>(Required) Specifies the name of the class. The name must begin with a letter; spaces and most non-alphanumeric characters are not allowed. The .cls extension will be appended automatically.</td>
</tr>
<tr>
<td>Final</td>
<td>If selected, specifies that inheritance from this class is disallowed. A <strong>FINAL</strong> option is included in the generated ABL code. A final class cannot be abstract.</td>
</tr>
<tr>
<td>Abstract</td>
<td>If selected, specifies that the class is abstract and cannot be instantiated. An abstract class is designed to serve as a super class from which other classes inherit and implement members. Therefore, an abstract class cannot be final.</td>
</tr>
<tr>
<td>Widget pool</td>
<td>If selected, specifies that a USE-WIDGET-POOL option will be included in the generated ABL code.</td>
</tr>
<tr>
<td>Serializable</td>
<td>If selected, marks the class as serializable by including the SERIALIZABLE modifier in the generated ABL code.</td>
</tr>
</tbody>
</table>

**Note:** If the class inherits from a super class that is not serializable, then it cannot be marked as serializable.

| Inherit       | Optionally specifies another class in the current project as a super class from which the new class inherits state and behavior. Click **Browse** or enter the class path by typing. A period (.) must separate each node from its parent folder in the path name. |

**Note:** The super class cannot be generic.
Lets you optionally specify one or more interfaces in the current project that the class implements. Click **Add** and select the desired interfaces at the Interface Selection dialog. Use the **Remove** button to remove an interface from the list after adding it.

**Note:** The class cannot implement a generic interface.

<table>
<thead>
<tr>
<th>Implements</th>
<th>Lets you optionally specify one or more interfaces in the current project that the class implements. Click <strong>Add</strong> and select the desired interfaces at the Interface Selection dialog. Use the <strong>Remove</strong> button to remove an interface from the list after adding it.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate default constructor</td>
<td>If checked, specifies that the class is to include a default constructor method.</td>
</tr>
<tr>
<td>Generate destructor</td>
<td>If checked, specifies that the class is to include a destructor method.</td>
</tr>
<tr>
<td>Generate super class constructors</td>
<td>If checked, specifies that the class is to include all constructor methods declared in the parent class.</td>
</tr>
<tr>
<td>Error-handling statement</td>
<td>Adds one of the following error-handling statements to the generated ABL code:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Block level:</strong> Includes the BLOCK-LEVEL ON ERROR UNDO, THROW statement in the generated ABL code</td>
</tr>
<tr>
<td></td>
<td>• <strong>Routine level:</strong> Includes the ROUTINE-LEVEL ON ERROR UNDO, THROW statement in the generated ABL code</td>
</tr>
<tr>
<td>Throw a Not Implemented exception</td>
<td>If selected, specifies that &quot;METHOD NOT IMPLEMENTED&quot; error code will be added to stubs for members implemented via interface and stubs for inherited abstract members.</td>
</tr>
<tr>
<td>Return a default value</td>
<td>If selected, specifies that default return values will be added to stubs for members implemented via interface and stubs for inherited abstract members.</td>
</tr>
<tr>
<td>Description</td>
<td>Lets you optionally enter a description for the class. This text will appear in the file header.</td>
</tr>
<tr>
<td>Purpose</td>
<td>Lets you optionally enter an explanation of the purpose of the class. This text will appear in the file header.</td>
</tr>
</tbody>
</table>

**See also**

- [Creating ABL source files](#) on page 381
- [Writing object-oriented code](#) on page 385

**New ABL Include wizard**

The **New ABL Include** wizard helps you create an ABL include file. This wizard appears when you select **File > New > ABL Include**.

The following controls are available:
Specifies a currently open project or a folder within the project where the include file is to be created. Click **Browse** if you want to select a project other than the current one (the default value).

<table>
<thead>
<tr>
<th>Container</th>
<th>Specifies the name of the include file. The .i extension is appended automatically.</th>
</tr>
</thead>
<tbody>
<tr>
<td>File name</td>
<td>Let you optionally enter a description for the file. This text will appear in the file header.</td>
</tr>
<tr>
<td>Description</td>
<td>Let you optionally enter an explanation of the purpose of the file. This text will appear in the file header.</td>
</tr>
<tr>
<td>Purpose</td>
<td>Let you optionally enter the author's name. This text will appear in the file header.</td>
</tr>
</tbody>
</table>

**See also**

- Creating ABL source files on page 381
- Writing procedural code on page 383

**New ABL Interface wizard**

The **New ABL Interface** wizard helps you create an interface file.

This wizard appears when you select **File > New > ABL Interface**.

The following controls are available:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Package root</strong></td>
<td>(Required) Specify a currently open project to contain the interface code and other project code. Click <strong>Browse</strong> if you want to select a project other than the current one (the default value).</td>
</tr>
<tr>
<td><strong>Package</strong></td>
<td>(Optional) Specify a package name, corresponding to a sub-folder of the package root, to contain the interface file. Click <strong>Browse</strong> or enter the folder path by typing. A period (.) must separate each subfolder from its parent folder in the path name.</td>
</tr>
<tr>
<td><strong>Interface name</strong></td>
<td>(Required) Specify the name of the interface. The name must begin with a letter; spaces and most non-alphanumeric characters are not allowed. The .cls extension will be appended automatically.</td>
</tr>
<tr>
<td><strong>Inherits</strong></td>
<td>(Optional) Specify one or more interfaces in the current project that the interface implements or inherits. Click <strong>Add</strong> and select the desired interfaces in the <strong>Interface Selection</strong> dialog. Use the <strong>Remove</strong> button to remove an interface from the list.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>(Optional) Enter a description for the interface. This text will appear in the file header.</td>
</tr>
<tr>
<td><strong>Purpose</strong></td>
<td>(Optional) Enter an explanation of the purpose of the interface. This text will appear in the file header.</td>
</tr>
</tbody>
</table>
New ABL Procedure wizard

The New ABL Procedure wizard helps you create a procedure file. This wizard appears when you select File > New > ABL Procedure.

The following controls are available:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Container</td>
<td>Specifies a currently open project or a folder within the project where the procedure file is to be created. Click Browse if you want to select a project other than the current one (the default value).</td>
</tr>
<tr>
<td>File name</td>
<td>(Required) Specifies the name of the procedure file. The .p extension is appended automatically.</td>
</tr>
<tr>
<td>Description</td>
<td>Lets you optionally enter a description for the file. This text will appear in the file header.</td>
</tr>
<tr>
<td>Purpose</td>
<td>Lets you optionally enter an explanation of the purpose of the file. This text will appear in the file header.</td>
</tr>
<tr>
<td>Author</td>
<td>Lets you optionally enter the author's name. This text will appear in the file header.</td>
</tr>
<tr>
<td>Error-handling statement</td>
<td>Adds one of the following error-handling statements to the generated ABL code:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Block level</strong>: Includes the BLOCK-LEVEL ON ERROR UNDO, THROW statement in the generated ABL code</td>
</tr>
<tr>
<td></td>
<td>• <strong>Routine level</strong>: Includes the ROUTINE-LEVEL ON ERROR UNDO, THROW statement in the generated ABL code</td>
</tr>
</tbody>
</table>

See also
ABL object-oriented extensions on page 365
Creating ABL source files on page 381
Adding internal procedures on page 384
Writing procedural code on page 383

New ABL Structured Include wizard

The New ABL Structured Include wizard helps you create an ABL include file with default AppBuilder markup. This wizard appears when you select File > New > Other > Progress OpenEdge > Editor > ABL Structured Include.

The following controls are available:
Specifies a currently open project or a folder within the project where the include file is to be created. Click **Browse** if you want to select a project other than the current one (the default value).

(Required) Specifies the name of the include file. The `.i` extension is appended automatically.

Lets you optionally enter a description for the file. This text will appear in the file header.

Lets you optionally enter an explanation of the purpose of the file. This text will appear in the file header.

Lets you optionally enter the author's name. This text will appear in the file header.

**See also**

- Creating new ABL source files on page 381
- Writing procedural code on page 383

**New ABL Structured Procedure wizard**

The **New ABL Structured Procedure** wizard helps you create an ABL procedure file with default AppBuilder markup.

This wizard appears when you select **File > New > Other > Progress OpenEdge > Editor > ABL Structured Procedure**.

The following controls are available:

<table>
<thead>
<tr>
<th>Container</th>
<th>Specifies a currently open project or a folder within the project where the procedure file is to be created. Click <strong>Browse</strong> if you want to select a project other than the current one (the default value).</th>
</tr>
</thead>
<tbody>
<tr>
<td>File name</td>
<td>(Required) Specifies the name of the include file. The <code>.p</code> extension is appended automatically.</td>
</tr>
<tr>
<td>Description</td>
<td>Lets you optionally enter a description for the file. This text will appear in the file header.</td>
</tr>
<tr>
<td>Purpose</td>
<td>Lets you optionally enter an explanation of the purpose of the file. This text will appear in the file header.</td>
</tr>
<tr>
<td>Author</td>
<td>Lets you optionally enter the author's name. This text will appear in the file header.</td>
</tr>
</tbody>
</table>
| Error-handling statement | Adds one of the following error-handling statements to the generated ABL code:  
  • **Block level**: Includes the BLOCK-LEVEL ON ERROR UNDO, THROW statement in the generated ABL code  
  • **Routine level**: Includes the ROUTINE-LEVEL ON ERROR UNDO, THROW statement in the generated ABL code |
Override/Implement Members wizard

The Override/Implement Members wizard helps you to:

• Locate inherited data members that you want to override.
• Satisfy the requirement to implement data members declared in interfaces implemented by the current class, and to implement inherited abstract members.

This wizard appears when you select Source > Override/Implement Members. The following controls are available:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tree view</td>
<td>Displays and lets you select data members in the class hierarchy that are eligible to be overridden or implemented.</td>
</tr>
<tr>
<td>Insertion position</td>
<td>Determines where the member code is inserted in the source file.</td>
</tr>
<tr>
<td>Generate comments</td>
<td>If checked, adds a comment block before the inserted member code.</td>
</tr>
<tr>
<td>Throw a Not Implemented exception</td>
<td>If selected, specifies that “METHOD NOT IMPLEMENTED” error code will be added to stubs for members implemented via interface and stubs for inherited abstract members.</td>
</tr>
<tr>
<td>Return default values</td>
<td>If selected, specifies that default return values will be added to stubs for members implemented via interface and stubs for inherited abstract members.</td>
</tr>
</tbody>
</table>

Generate ABLDoc wizard

The Generate ABLDoc wizard enables you to generate an ABLDoc documentation for a selected project.

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document title</td>
<td>Specifies the name as the title for the generated ABLDoc documentation.</td>
</tr>
<tr>
<td>Destination</td>
<td>Specifies a default directory to save all output files.</td>
</tr>
<tr>
<td>Select the files for which you want to generate the ABLDoc</td>
<td>Specifies the project and the corresponding files for which you want to generate the ABLDoc documentation.</td>
</tr>
<tr>
<td>Create ABLDoc for members with access mode</td>
<td>Specifies if you want to include members with visibility other than public.</td>
</tr>
</tbody>
</table>

See also

Creating ABL source files on page 381
Writing procedural code on page 383
Coding members implemented via interface or abstraction on page 389
Writing procedural code on page 383
Profiler Editor

The Profiler Editor opens when you profile a session in the launch configuration or you click a .prof file. It displays the details of the session for which you created the .prof file.

Module Details tab

The Module Details tab of the Profiler Editor displays the execution time of the modules in the current session. It displays the modules that are calling other modules and also the modules that have been called.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module Name</td>
<td>Displays the name of the module.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The row displays the module name and name of the file that contains the module separated by a space, for example, Proc1 C:ex/Procedure.p, where Proc1 is the name of the module and C:ex/Procedure.p is the name of the file.</td>
</tr>
<tr>
<td>Times Called</td>
<td>Displays the number of times the module has been called.</td>
</tr>
<tr>
<td>Average Time Per call (sec)</td>
<td>Displays the average of the total time taken to execute and the number of times the module was called.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Displays in seconds with microsecond precision.</td>
</tr>
<tr>
<td>Total Time (sec)</td>
<td>Displays the total time taken by this module to execute.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Displays in seconds with microsecond precision.</td>
</tr>
<tr>
<td>% of Session</td>
<td>Displays the percentage of session that is consumed by the execution of this module.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The percentage is presented numerically and graphically.</td>
</tr>
</tbody>
</table>

You can select a module in the table above and view the details of the modules that called it or the modules that this module called.
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calling Module</td>
<td>Displays the names of the modules that are being called by the module whose details you are viewing.</td>
</tr>
<tr>
<td>Times Calling</td>
<td>Displays the number of times the module was called.</td>
</tr>
<tr>
<td>% of Session</td>
<td>Displays the percentage of the session that is consumed when this module is calling the module selected in the Calling Modules column.</td>
</tr>
<tr>
<td>Called Modules</td>
<td>Displays the name of the module that is called by the module whose details you are viewing.</td>
</tr>
<tr>
<td>Times Calling</td>
<td>Displays the number of times the module was called.</td>
</tr>
<tr>
<td>% of Session</td>
<td>Displays the percentage of the session that is consumed when this module is calling the module selected in the Called Modules column.</td>
</tr>
</tbody>
</table>

**AVM Information tab**

The AVM Information tab of the Profiler Editor displays the general information about the session and the databases being used by the profiled session.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Id</td>
<td>Displays the user Id that profiled the session.</td>
</tr>
<tr>
<td>Date of profiling</td>
<td>Displays the date and the time when the session was profiled.</td>
</tr>
<tr>
<td>Description</td>
<td>Displays the description (-Description) that you provided when you enabled the profiling in the Profiler tab in the Launch Configuration or added using the -Description option from the command line.</td>
</tr>
<tr>
<td>Parameters</td>
<td>Displays the parameters that the session passed to the AVM.</td>
</tr>
<tr>
<td>Database name</td>
<td>Displays the names of the databases that were used by the session.</td>
</tr>
<tr>
<td>Number of tables</td>
<td>Displays the number of tables in the selected database.</td>
</tr>
</tbody>
</table>

**Call Tree tab**

The Call Tree tab of the Profiler Editor displays the call tree hierarchy of the session. It enables you to evaluate the calls that consume most of the session.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Call Order</td>
<td>Displays the order in which the calls were made to the selected module.</td>
</tr>
<tr>
<td>Times Called</td>
<td>Displays the number of times the module called.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Cumulative Time (secs)</td>
<td>Displays the total time taken to execute the call including the time taken to execute the subcalls.</td>
</tr>
<tr>
<td>% of Session</td>
<td>Displays the percentage of the session consumed by the call.</td>
</tr>
</tbody>
</table>

**Note:** The details on the Call Tree tab are displayed only if the session was profiled using the Tracing-Filter "*" option.

## ABL Scratchpad view

**ABL Scratchpad** is a multi-tabbed view provided by Progress Developer Studio for OpenEdge. It allows you to write and test the ABL code without having to save it or creating a new file or project. By default, the shared AVM runtime is used for the execution of the code. However, it provides the option of selecting the runtime of any existing OpenEdge projects for execution. This saves time and avoids the need to create a project or document multiple times for testing the code in different runtimes.

**Note:** The **ABL Scratchpad** view appears in all the OpenEdge perspectives except the **DB Navigator** perspective. By default, the **ABL Scratchpad** view appears at the bottom of the page in all the perspectives.

To access the **ABL Scratchpad** view, select **Window > Show View > ABL Scratchpad** from the main menu or use the keyboard shortcut **CTRL+F9**.

The **ABL Scratchpad** view toolbar contains the following options:

| **Open New Tab** | Opens a new tab in the **ABL Scratchpad** view. You can also access this option from the **ABL Scratchpad** view context menu or by using the **CTRL+T** keyboard shortcut. The keyboard shortcut for closing a tab is **CTRL+ALT+T**. |
| **Check Syntax**  | Verifies the correctness of the code. It checks the syntax of the ABL code written in the **ABL Scratchpad** view and displays a message stating whether the syntax is correct or not. You can also access this option from the **ABL Scratchpad** view context menu or by using the **CTRL+SHIFT+C** keyboard shortcut. If the syntax of the code is incorrect, an error is displayed stating the position of the error. |
| **Run**           | Executes the ABL code either in a new AVM or the project’s AVM. These AVM options are provided in the drop-down menu of the Run button. You can also access this option from the **ABL Scratchpad** view context menu. To launch in a new AVM, use the **ALT+SHIFT+F11** keyboard shortcut and to launch using the project runtime, use the **ALT+F11** keyboard shortcut. When you execute the code by directly clicking the Run button, the **ABL Scratchpad** view uses the most recently selected AVM to run the code. If there are any errors while executing the code, an error dialog is displayed. |
**Save As**
Allows you to save the code in the **ABL Scratchpad** view as a physical file. You can also access this option from the **ABL Scratchpad** view context menu or by using the **CTRL+ALT+S** keyboard shortcut.

On invoking this action, the Save As dialog appears. The .sharedavm folder, present in the workspace, is the default location where the file is saved, if it is executed in a shared AVM. If the code is executed in any other OpenEdge project's runtime, then it is saved in the corresponding project's folder in the workspace. However, you can save the file at any location that you like. The default extension in which the file is saved is .p. However, you can choose to save it any format that you like.

If the view is closed and reopened for the same session, the tabs and the data present persist. The AVMs selected for each tab also persist. The same applies when you restart Progress Developer Studio for OpenEdge.

**Clear**
Clears the entire text present in the **ABL Scratchpad** view.

You can also access this option from the **ABL Scratchpad** view context menu or by using the **CTRL+Q** keyboard shortcut.

**AVM selection**
Specifies the runtime for executing the code in the **ABL Scratchpad** view.

The shared AVM runtime is used as the default runtime for executing the code in the **ABL Scratchpad** view. However, you can use this drop-down menu to select any OpenEdge project's runtime (if present) in the workspace.

Depending on the runtime selected and the database connection associated with that runtime, various features such as color coding, content assist, etc., are provided in the scratchpad. These changes happen dynamically as the runtime changes. Since it is a multi-tabbed view, the runtime selected for each tab is recorded. As you shift across tabs, the runtime changes reflect accordingly.

In addition to the above options, the **ABL Scratchpad** view provides the following features:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cut, Copy, and Paste</td>
<td>Implements the general Cut, Copy, and Paste commands. These options are available in the <strong>ABL Scratchpad</strong> view context menu and as shortcut key commands.</td>
</tr>
<tr>
<td>Content assistance</td>
<td>Helps you write code faster and more efficiently. It provides a list of accessible keywords such as variable, methods, data types, etc. This feature is similar to that of the corresponding functionality in ABL Editor. It can be invoked with <strong>CTRL+SPACE</strong> key bindings. If the selected runtime is connected to a database, then the <strong>ABL Scratchpad</strong> view displays the list of tables present in that database on invoking the <strong>CTRL+SPACE</strong> command.</td>
</tr>
<tr>
<td>Color coding</td>
<td>Differentiates the keywords present in the ABL programming language from the rest of the words, by representing them with different colors that are highlighted. The selection of runtime affects the color coding schema. Color coding in the <strong>ABL Scratchpad</strong> view is similar to the ABL Editor color coding schema.</td>
</tr>
</tbody>
</table>
Hot keys

Hot keys are useful for frequently used coding functions. The table below lists some of the Editor's default hot keys.

<table>
<thead>
<tr>
<th>Function</th>
<th>Keyboard sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Constructor</td>
<td>ALT+SHIFT+C</td>
</tr>
<tr>
<td>Add Constructors from Super Class</td>
<td>ALT+SHIFT+S</td>
</tr>
<tr>
<td>Add Destructor</td>
<td>ALT+SHIFT+R</td>
</tr>
<tr>
<td>Add Function</td>
<td>ALT+SHIFT+F</td>
</tr>
<tr>
<td>Add Method</td>
<td>ALT+SHIFT+M</td>
</tr>
<tr>
<td>Add Procedure</td>
<td>ALT+SHIFT+P</td>
</tr>
<tr>
<td>Add Property</td>
<td>ALT+SHIFT+Y</td>
</tr>
<tr>
<td>Add Static Constructor</td>
<td>ALT+SHIFT+T</td>
</tr>
<tr>
<td>Check Syntax</td>
<td>CTRL+SHIFT+C</td>
</tr>
<tr>
<td>Code-Completion Assistance</td>
<td>CTRL+SPACE</td>
</tr>
<tr>
<td>Correct Case</td>
<td>CTRL+SHIFT+F</td>
</tr>
<tr>
<td>Expand Keywords</td>
<td>CTRL+SHIFT+1</td>
</tr>
<tr>
<td>Fix Indentation</td>
<td>CTRL+I</td>
</tr>
<tr>
<td>Fix Prototypes</td>
<td>CTRL+SHIFT+2</td>
</tr>
<tr>
<td>Go to Matching Element</td>
<td>CTRL+SHIFT+P</td>
</tr>
<tr>
<td>Next Member</td>
<td>CTRL+SHIFT+DOWN</td>
</tr>
<tr>
<td>Open Declaration</td>
<td>F3</td>
</tr>
<tr>
<td>Override/Implement Methods</td>
<td>ALT+SHIFT+O</td>
</tr>
<tr>
<td>Previous Member</td>
<td>CTRL+SHIFT+UP</td>
</tr>
<tr>
<td>Keyword help</td>
<td>SHIFT+F2</td>
</tr>
<tr>
<td>Run Program</td>
<td>CTRL+U</td>
</tr>
<tr>
<td>Find references</td>
<td>CTRL+SHIFT+G</td>
</tr>
<tr>
<td>Surround With Catch Block</td>
<td>ALT+SHIFT+Z, C</td>
</tr>
</tbody>
</table>
See also
Writing ABL code on page 381

**Line Summary view**

_**Line Summary**_ view displays the module file of the module and the line details of the file that was called.

When you open a .prof file, _Line Summary_ view opens. You can also select **Window > Show View > Profiler > Line Summary** from the main menu.

The right section of the _Line Summary_ view displays the debug listing content in the right section, only if you selected the **Generate Debug Listing files** check box in the **Profiler** tab of the Run Configuration. The left section displays the following:

<table>
<thead>
<tr>
<th>Line Number</th>
<th>Displays the line number in the module file that was called.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Times Called</td>
<td>Displays the number of times the line number in the module file that was called.</td>
</tr>
<tr>
<td>Average Time (secs)</td>
<td>Displays the average time taken by the line to execute calls.</td>
</tr>
<tr>
<td>Total Time (secs)</td>
<td>Displays the total time taken by the line to execute calls.</td>
</tr>
</tbody>
</table>
Introducing the Progress Developer Studio for OpenEdge Visual Designer

The OpenEdge Visual Designer is an editor that helps you develop attractive, powerful graphical user interfaces (GUIs) for ABL applications. Visual Designer users can take advantage of the rich functionality and the look and feel of .NET controls without using any non-ABL language or leaving the Progress Developer Studio for OpenEdge environment.

Note: The Visual Designer is supported only on Windows platforms.

UI visualization and ABL code
With the Visual Designer, you build OpenEdge GUI for .NET forms and dialogs, also referred to as visual containers, by using familiar GUI techniques to place and arrange controls on a Design Canvas. Behind the scenes, the Visual Designer takes care of generating the corresponding ABL code that allows the application to display these .NET forms and controls at run time. This ABL code, in the form of a class (.cls) file, is immediately available in the standard ABL Editor, where you can inspect it and add appropriate business logic. Progress Developer Studio for OpenEdge maintains synchronization between the class file and the graphical display of the UI model on the Design Canvas.

Property and event management
When an object is selected on the Design Canvas, the Properties view displays all properties and events associated with the object and allows you to edit their values.

The Visual Designer simplifies the process of creating event subscriptions and adding the appropriate event logic. For example, suppose you have placed a button on a form and want to define how the application should respond when a user clicks that button. When you simply double-click the event name in the Properties view:
The Visual Designer automatically adds to the class file both a subscription to a button-click event and a method to contain the code defining the application response to the event.

The ABL Editor displays the class file with the cursor positioned at the newly created event method.

**Custom user-defined controls**

In addition to the extensive set of commercially published UI controls that the Visual Designer provides, you can create and reuse custom controls that extend or combine existing controls. You can create two types of custom controls:

- **Inherited controls**, which derive properties and events from a super class (that is, a parent control).
- **User controls**, which are composite controls made up of multiple individual controls grouped in a container.

**Demos**

| Building a Navigation Panel as an ABL User Control |
| Creating a Data-bound Treeview as an Inherited Control (Part 1) |
| Creating a Data-bound Treeview as an Inherited Control (Part 2) |
| Defining Event Subscriptions and Event Handlers (Part 1) |
| Defining Event Subscriptions and Event Handlers (Part 2) |
| Adding Field-level Controls to a Form |

**Note:** The OpenEdge Release 10.2B demos have not been updated for OpenEdge Release 11.0. Since 10.2B, OpenEdge Architect has been renamed as Progress Developer Studio for OpenEdge. In addition, OpenEdge Release 11.0 includes some feature enhancements and other changes. Despite the differences between 10.2B and 11.0, the demos are still a useful introduction to Progress Developer Studio for OpenEdge.

For details, see the following topics:

- Concepts
- Tasks
- Reference

**Concepts**

**Benefits of the Visual Designer**

Among the principal benefits that the Visual Designer provides to ABL application developers are the following:

- The OpenEdge GUI for .NET (that is, the application presentation layer developed with the Visual Designer) takes advantage of full-featured, visually compelling .NET controls.
- The Visual Designer supports coding entirely in ABL. There is no need to learn or use any other programming language.
• Controls can be bound to ABL data sources (databases, temp tables, and ProDataSets).

• Use of the Visual Designer facilitates separation of the application presentation layer (UI) from the business logic layer, the recommended approach for promoting efficient maintenance and reuse of application modules.

See also
UI controls on page 477
The Visual Designer tool set on page 480
Developing a visual container on page 482
Introducing the Progress Developer Studio for OpenEdge Visual Designer on page 471

Code associated with a Visual Designer component

The Visual Designer generates and maintains one class (.cls) file for each visual container that you design. This class contains the code needed to allow the ABL Virtual Machine (AVM) to display and enable the visual container. The code includes:

• A constructor and a destructor

• An InitializeComponent method that includes variable value assignments, property settings, and event subscriptions associated with the visual container

• An event method for each subscription

• A DoWait method that suspends program execution pending user action

Progress Developer Studio for OpenEdge synchronizes the ABL code with the Visual Designer display, so that changes made in either editor are immediately reflected in the other. However, it is recommended that you use the Visual Designer tools for all changes to the UI, and edit only application logic (for example, event logic) in the ABL Editor. In this way you can reduce the chance of introducing errors in the UI code.

See also
Viewing and editing the container's source code on page 484

.NET assemblies and the Visual Designer

.NET classes, including the forms and controls used to build an OpenEdge GUI for .NET, reside in collections called assemblies. An application that includes .NET classes must have a list of the assemblies in which those classes are stored. This list of references is contained in a file called assemblies.xml.

An assemblies.xml file is automatically created for your Progress Developer Studio for OpenEdge project when you place a control on a form or manually add an assembly to the project. As you use .NET objects to build UI components, Progress Developer Studio for OpenEdge adds the appropriate assembly references to the file so that the assemblies.xml file for a project contains all references needed for that project.
In addition to maintaining the `assemblies.xml` file, Progress Developer Studio for OpenEdge also adds a Referenced Assemblies node to the Project Explorer view, providing a convenient list of the assemblies in use in the project. If your application uses .NET classes other than those you access with the Visual Designer, you can add them to the `assemblies.xml` file through a dialog that is accessible via right-clicking in the Project Explorer view (or via the Progress OpenEdge Properties page).

**Note:** When you delete a control with the Visual Designer, the assembly reference is not automatically removed from the `assemblies.xml` file or from the Project Explorer view. Although unused assembly references do no harm, if you want to delete them, you must do so manually. Use the Assemblies dialog in project properties (Project > Properties > OpenEdge > Progress > Assemblies) to do this.

Your application must have access to the assemblies at run time. Thus, the run-time environment must include all referenced assemblies.

### Location of `assemblies.xml` and controls

By default, Progress Developer Studio for OpenEdge creates the `assemblies.xml` file in the project's root folder. To specify a different location, add `-assemblies directory-path` to the (Project > Properties > Progress OpenEdge). You can enter the absolute directory path or the path relative to the project working directory.

All controls installed with Progress Developer Studio for OpenEdge are automatically registered in the global assembly cache (GAC), the standard repository for .NET assemblies that are designed for use by multiple applications running on the local machine. The GAC enables applications to locate and use all of the assemblies that it contains. Any assemblies used by the application that are not registered in the GAC must reside in the same directory as the `assemblies.xml` file. If you use such assemblies, be sure to copy them to that location. This requirement applies both to the run-time environment and to the design-time environment. Failure to keep the `assemblies.xml` file and the assemblies together in the same directory prevents applications from loading the controls at run time, and may cause various design-time issues.

**Note:** For more information about considerations related to .NET assemblies, see *OpenEdge Deployment: Managing ABL Applications*.

### Updating assembly references to use new controls

If you install a new version of an assembly, it is necessary to update existing assembly references in order to use the new controls in applications that you developed previously. OpenEdge provides an Update Assembly References tool to help you accomplish this. The tool runs independently outside of the Eclipse framework, but you can launch it from Progress Developer Studio for OpenEdge by means of an option on the OpenEdge > Migration menu.

**Note:** For more information, see Updating assembly references on page 516.

### Default assemblies

For any ABL application that accesses .NET objects, the AVM automatically loads certain .NET assemblies, including the appropriate versions of:

- **Progress.NetUI.dll** - Assembly where all custom OpenEdge .NET classes reside
- **Mscorlib.dll** - Assembly where all core Microsoft .NET classes reside
- **System.Windows.Forms.dll** - Assembly where all Microsoft form and control classes reside
- **System.Drawing.dll** - Assembly where all Microsoft graphics classes reside
ABL can therefore locate any .NET type that you reference from these assemblies without their being listed in the assemblies.xml file.

**See also**
- UI controls on page 477
- Working with .NET resources on page 515
- Adding and removing assembly references on page 516
- Updating assembly references on page 516

**Visual container types**

As the starting point for building an OpenEdge GUI for .NET, you create a visual container by using one of three Progress Developer Studio for OpenEdge wizards: **New ABL Form**, **New ABL Dialog**, or **New ABL MDI Form**.

**ABL Form**

The **New ABL Form** wizard creates a blank form.

![New ABL Form](image)

**ABL Dialog**

The **New ABL dialog** wizard creates a simple dialog with standard **OK** and **Cancel** buttons. An event subscription and event logic for a click are pre-coded for each of these buttons.
ABL MDI Form

The **New ABL MDI form** wizard creates a form enabled for multiple-document interface functionality, meaning that it can be the parent of child forms. The form includes a menu strip and a toolbar containing common menus (File, Edit, View, Tools, Windows, Help) and command buttons (New, Open, Save, Print, Print Preview, and Help) with pre-coded event subscriptions and logic. It also includes a status bar at the bottom.

The menu strip, toolbar, and status bar have special design-time behavior: Any of these default container objects, or any of the controls on them, when clicked, launches an embedded tool that provides an easy way to add new controls. Note that these tools do not appear on the form at run time.
Note: It is possible to embed an ABL application window, such as a window developed with OpenEdge AppBuilder, in any form. For example, you can create an MDI form that includes an ABL window as a child. However, you must manually write the ABL code to do this. The Visual Designer does not support programming tasks related to ABL windows. For instructions on embedding windows in forms, see OpenEdge Development: GUI for .NET Programming.

See also
- Developing a visual container on page 482
- OpenEdge GUI for .NET wizards on page 519

Implementing the OpenEdge GUI for .NET in your application

To implement the UI in your application, you write ABL code that instantiates the classes in which the visual containers are defined. You can add application logic to the class files generated by the Visual Designer where it suits your purposes to do so. In general, however, it is good practice to write separate procedures that are called by UI events. This use of separate layers for business logic and UI can make it easier to maintain and reuse your application code.

See also
- Developing a visual container on page 482

Application run-time environment

All .NET assemblies used by the application must be available at run time, either in the Windows Global Assembly Cache (GAC) or in the Progress\OpenEdge\bin directory. Therefore, when you set up your run-time environment, you need to ensure that the required assemblies are installed on the end user’s machine. In general, the standard Microsoft forms and controls are installed with Windows. OpenEdge Ultra Controls for .NET, however, typically will need to be installed as part of application setup.

The development license for the OpenEdge Ultra Controls for .NET entitles application developers to distribute the controls freely.

See also
- .NET assemblies and the Visual Designer on page 473
- Working with .NET resources on page 515

UI controls

All Progress Developer Studio for OpenEdge users have access to a standard set of Microsoft .NET controls. These standard controls are included with the Microsoft .NET Framework, which the Progress Developer Studio for OpenEdge installation program installs on the host machine if it is not already present.

A separately licensed option allows the use and redistribution of OpenEdge Ultra Controls for .NET, developed by Infragistics, which offer enhanced functionality and ease of use. The set of Progress-supported controls available on your system depends on whether the OpenEdge Ultra Controls for .NET option is installed.
Progress Developer Studio for OpenEdge also installs a trial version of Telerik UI for WinForms by default. It includes controls that you can use to build your business applications. If you need a different version of Telerik, you can download it from the Telerik web site.

**Note:** If you need help on Microsoft, Telerik, or Infragistics controls, press F1. The Microsoft website, Telerik web site or Infragistics web site page opens.

In addition, there may be other third-party controls installed in the development environment. Progress warrants that all .NET-compliant controls can be loaded in the Visual Designer and implemented on OpenEdge GUI for .NET forms. However, with respect to design and function, Progress supports only the set of controls that is enabled by default in the Visual Designer.

**Visual and non-visual controls**

UI controls fall in one of two basic categories: visual or non-visual.

- Visual controls appear on the application's user interface and generally allow user interaction, display data, or both. Examples include buttons, combo boxes, and datagrids.

- Non-visual controls do not appear on the user interface at run time; instead, they support visual controls or provide other services such as logging. Non-visual controls typically hold data, formatting, or other information needed by one or more interactive controls. An example is the ProBindingSource, which serves as an intermediary between an actual ABL data source and the control, such as a grid, that displays the data.

Visual controls can be placed and manipulated on the Design Canvas. Non-visual controls appear in a separate area at the bottom of the Design Canvas; you can select them for the purpose of setting their properties, but since they have no visual characteristics, you cannot resize or format them.

**Note:** In the context of .NET development, non-visual controls are often referred to as components.

**See also**

- The Visual Designer tool set on page 480
- Working with UI controls on page 485

**Container controls**

Some controls are designed as container controls; that is, they can function as the parent to a set of child controls. Typically, child controls are visually contained within the container on a finished UI form. At design time, you simply place the desired child controls on the container.

Container controls appear in the Containers group in the default Toolbox configuration. They are also identified in the control summary.

**See also**

- Developing a visual container on page 482
- Creating a user control on page 488

**Data-bound controls**

Certain visual controls are capable of displaying data retrieved from a database connected to the application. The Visual Designer simplifies the task of binding the data to the control.
Three objects are required:

- A data-displaying visual control
- A ProBindingSource non-visual control
- An ABL data object (query, temp table, or ProDataSet)

The data object is defined with standard ABL constructs; no special coding is necessary.

See also

Linking controls to data on page 502

Custom user-defined controls

You can define two types of custom controls:

- **User control** - A collection of individual controls in a container. For example, you might create a user control called "Address" that includes text box controls for user entry of name, street address, and city; a combo box control that allows the user to select the state; and accompanying label controls that identify these fields.

- **Inherited control** - A control derived from another control. You can set properties and events as appropriate for your purposes. For example, you might created an inherited control called ContactButton (derived from a standard button control) for which you specify the size and color, the text "Contact," and event logic that responds to a button click by launching a pre-addressed e-mail message.

After you create a custom control, you can add it to your Visual Designer Toolbox to make it available for use on any forms you design.

Note: If any of your projects use the global Toolbox, be sure that each such project has all of the other projects on its PROPATH so that custom controls that you add to the Toolbox are available in all projects.

See also

Creating a user control on page 488
Creating an inherited control on page 489
New ABL User Control wizard on page 524
New ABL Inherited Control wizard on page 526

Control properties

Each UI object has a number of properties that define its appearance and behavior. The Visual Designer lets you set these properties in several ways:

- Size and position can be set by clicking and dragging objects on the Design Canvas.
- Menus provide commands for setting alignment, spacing, front-to-back order, and tab order.
- The Properties view lets you edit all available properties of selected objects.

The set of available properties depends on the control. Many controls have built-in wizards or other design aids tailored to the characteristics of that specific control.
See also
Defining properties and appearance on page 493
Visual Designer Properties view on page 530

Event behavior

Every control has a set of events that it raises in response to specific user actions or system occurrences. The list of a control’s events appears on the **Events** tab in the **Properties view** when the control is selected in the Visual Designer.

To make your application respond to a given UI event, such as a button click, your ABL code must both subscribe to the event and define the desired response. Simply double-clicking an event in the Properties view automatically generates an event subscription and an event method in which you can define the appropriate logic.

See also
Coding event logic on page 510
Visual Designer Properties view on page 530

Access to controls

You access the available set of controls through the **Toolbox**.

You can customize the **Toolbox** to suit your preferences, arranging the controls in any groups you like. You can also modify the available set by adding or deleting controls.

---

**Note:** Progress Software Corporation supports only those controls that are installed with Progress Developer Studio for OpenEdge.

---

See also
Visual Designer components on page 480
Using the Toolbox on page 485
Sharing Toolbox settings on page 512
Visual Designer Editor on page 529

The Visual Designer tool set

This section describes the components of the Visual Designer and related tools.

**Visual Designer components**

The major components of the Visual Designer are the Toolbox, Design Canvas, Properties view, and Outline view.

**Toolbox**

The Toolbox is a list of controls that are available for use in the UI. You select controls that you want to place on the form from this list. You can customize the set of available controls and arrange them in logical groups that you define. Customized settings can be shared with other users or copied to other projects.
By default, each OpenEdge project you create uses a dedicated Toolbox that is specific to that project. Alternatively, you have the option of using the global Toolbox, which is shared among all or a subset of the projects in the workspace. You choose whether to use the global Toolbox when you create a project. For an existing project, you can view and change the setting on the Progress OpenEdge properties page.

Customizations that you apply while working in a project that uses the global Toolbox affect the Toolbox for all other such projects. For any projects that do not use the global Toolbox, customizations are saved on a per-project basis, enabling you to tailor the Toolbox to the particular needs of different projects.

**Note:** The PROPATH for every project that uses the global Toolbox should include the project directories of all other projects that share the Toolbox. If these projects also share an AVM instance, Progress Developer Studio for OpenEdge automatically updates the PROPATH and takes care of this requirement. However, for any project that does not use the shared AVM option, be sure to manually add the appropriate directories to the PROPATH for each project.

**Design Canvas**

The Design Canvas is the editing area in which you model the appearance of the UI. It provides an accurate visualization of how the window will appear to the user at run time, with the exception that it does not display actual data. You use the mouse to size objects by dragging their borders, and to position controls on the form.

**Properties view**

While a Visual Designer editing window has focus, the Properties view includes both a Properties tab and an Events tab, where you can view and edit the visual and behavioral characteristics of the object or objects that are selected on the Design Canvas.

**Outline view**

While a Visual Designer editing window has focus, the Outline view shows each component of the design as a tree node. This hierarchical representation of the form or user control complements the graphical display on the Design Canvas. You can use the Outline view as an alternative means of selecting, copying, deleting, renaming, and re-parenting controls.

**See also**

Developing a visual container on page 482  
Working with UI controls on page 485  
Customizing the Visual Designer Toolbox on page 511  
Using the Outline view on page 494  
Workspace components on page 529

**The Visual Designer perspective**

The Visual Designer perspective gives you a handy way to set up your workspace with maximum space allotted to the tools that you typically use most while working on an OpenEdge GUI for .NET project. It includes a large editing area, which displays the Toolbox and Design Canvas along with any other open file buffers (such as the ABL Editor tab for the class source code). It also includes a window with tabs for the Properties view and the Resources view. This window is vertically maximized to display as many control properties as possible without requiring scrolling.

In addition, the default Visual Designer perspective includes a Fast View shortcut to open the Class Browser.

**See also**

Developing a visual container on page 482
Additional tools

Progress Developer Studio for OpenEdge tools that are not integral to the Visual Designer, but have particular value while you are building an OpenEdge GUI for .NET, include:

- **Class Browser** - The Class Browser offers a tree view that provides detailed information on the classes and .NET assemblies defined in or referred to by your projects.
- **Meta Catalog** - The Meta Catalog enables you to create a comprehensive index of the code base in both your local workspace and your team’s shared source code control system.
- **ABL Editor** - The standard OpenEdge code editor is tightly linked to the Visual Designer to allow you to inspect and edit the source code for design classes and related procedures.
- **Outline view** - When a Visual Designer editor buffer has focus, the **Outline view** provides a hierarchical list of the form and its contents, allowing you to select, copy, re-parent, and delete controls.
- **Debugger** - The Debugger provides a full set of tools for testing and debugging ABL code.

See also

*Using the Outline view* on page 494

Tasks

Developing a visual container

Creating the container

To create a new visual container class:

1. Select **File > New** and then the type of container you want: **ABL Form**, **ABL Dialog**, or **ABL MDI Form**. The **OpenEdge GUI for .NET** wizard for the selected type appears.

2. Enter the appropriate values as described in the **OpenEdge GUI for .NET wizards** reference help. Only the **Package root** and the **Form name** or **Dialog name** fields are required. (The name must begin with a letter; spaces and most non-alphanumeric characters are not allowed.)

3. After entering the desired values, click **OK**. Progress Developer Studio for OpenEdge creates a new class file with the same name that you assigned to the new form and opens it in a Visual Designer editing window. The **Design Canvas** displays the form with its default content, if any.

**Note:** Do not change the name of the class file. The class will not compile if its name and that of the file do not match.

4. You can now proceed to size the container and add controls to it.
Note: It is possible to embed an ABL application window, such as a window developed with OpenEdge AppBuilder, in any form. For example, you can create an MDI form that includes an ABL window as a child. However, you must manually write the ABL code to do this. The Visual Designer does not support programming tasks related to ABL windows. For instructions on embedding windows in forms, see OpenEdge Development: GUI for .NET Programming.

See also
Visual container types on page 475
Sizing the container on page 483
Working with UI controls on page 485
Creating custom controls on page 488
OpenEdge GUI for .NET wizards on page 519

Opening an existing container
To open an existing UI design class, right-click the file name in the Project Explorer view and select Open With. Then, from the sub-menu, select OpenEdge ABL Editor to display the source code, or OpenEdge Visual Designer to display the design.

You can also simply double-click the file name in the Project Explorer view. Progress Developer Studio for OpenEdge opens the form either in a Visual Designer editing window or in the ABL Editor, whichever you last selected when opening the file with the Open With command. If you have not previously used Open With for the file, it opens in the Visual Designer.

Note: You cannot open multiple instances of the same file in the Visual Designer. However, multiple instances of the file in the OpenEdge ABL Editor (via the New Editor command) are allowed.

See also
Sizing the container on page 483
Working with UI controls on page 485

Sizing the container
You can control or define the size of the form in either of two ways:

1. Click and drag any of the three handles located on the right and bottom edges and the lower right corner of the form, which always remains pinned to the top left corner of the Design Canvas.

2. To set precise dimensions in pixels, enter the desired values for the Size property in the Properties view.

Note: The Minimize, Maximize, and Close buttons that appear by default in the window control box at the top right corner of the form are run-time features; they are not functional at design time. You can enlarge or maximize the editing window if you need more space for the Design Canvas. You can also hide the Toolbox.

See also
Working with UI controls on page 485
**Viewing and editing the container's source code**

To display the source code in the OpenEdge ABL Editor while editing a design class in the Visual Designer, do one of the following:

1. Right-click in the **Design Canvas** and select **View Source** from the context menu.
2. Select **Design > View Source**.
3. Press **F9**.

   All features of the ABL Editor are available.

Conversely, to display the form in the Visual Designer while editing the source code, do one of the following:

1. Right-click in the **Design Canvas** and select **View Design** from the context menu.
2. Select **Design > View Design**.
3. Press **Shift+F9**.

Changes are synchronized when you switch between the design view and the source view, and when you save the file. Thus, changes that you make with either editor are immediately reflected in the other.

Once you have opened the class file in either the Visual Designer or the ABL Editor, it remains open in that editor until you close it.

**Note:** When editing the ABL code, be careful not to introduce errors in the UI code. It is advisable to use the Visual Designer for all changes to the UI, and edit only application logic (for example, event logic) in the ABL Editor. In particular, do not modify the InitializeComponent method created by the Visual Designer; in doing so, you might cause an error that would not prevent the code from compiling, but would make it impossible for the Visual Designer to open the file.

**See also**

- [Code associated with a Visual Designer component](#) on page 473
- [Coding event logic](#) on page 510

**Deleting a container**

To delete a container class, select the .cls file in the **Project Explorer** view and press **Delete**. You cannot delete a class via the Visual Designer, and you cannot delete the container object (the form) from the Design Canvas.

When you delete a UI .cls file, the corresponding compiled r-code file is also deleted.

**Note:** For some UI classes that you create, an accompanying RESX file of the same name is created. This file is not automatically deleted when you delete a class file, so you should manually clean up unneeded RESX files. By default, the Resources view does not list RESX files. If you want to see them in the Resources view, click ➔ select **Filters**, and uncheck the **.resx entry**.

**See also**

- [Deleting controls from a container](#) on page 493
Working with UI controls

Using the Toolbox
You use the Toolbox as the source of all controls that you incorporate in an OpenEdge GUI for .NET design.

By default, the Toolbox is docked to the right edge of the editing area. If you wish, you can change the docking position through the Visual Designer Preferences page.

Opening and closing Toolbox sections
Controls in the Toolbox are arranged in control groups:

![Toolbox control groups](image)

To open or close a control group, click its title bar. You can also right-click and select Expand All Control Groups from the context menu. You can have multiple control groups open simultaneously.

**Note:** You can customize the control groups and controls that appear in the Toolbox. In particular, if you create custom controls, you need to add them to the Toolbox to make them available for use. See Customizing the Visual Designer Toolbox.

Placement mode and selection mode
You select a control in the Toolbox by clicking its name. While a single control is selected in the Toolbox, the Visual Designer is in placement mode, meaning that it is ready to place an instance of that control on the Design Canvas. When the cursor hovers over the form in placement mode, it appears as blue crosshairs with the icon for the selected control (for example, with the ProBindingSource control selected: ![ProBindingSource icon](image)). In this mode, clicking or dragging on the form results in adding a control. Note that you can also perform Toolbox customization operations while in placement mode.

To operate on the form or on controls that have already been placed on the form, you must be in selection mode. To put the Toolbox in selection mode, click the pointer tool (highlighted in the preceding illustration).

When the cursor hovers over the form in selection mode, it appears as an arrow pointer. In this mode, you can select one or more objects on the Design Canvas and move them, edit their properties, or delete them.

**Sticky placement mode**
By default, the Toolbox automatically switches from placement mode to selection mode after you place a control on the Design Canvas. When you want to place several instances of a control in succession, you can set the Toolbox to remain in placement mode until you are ready to revert back to the default behavior.

To lock the Toolbox in placement mode, right-click a control in the Toolbox and select Make Sticky from the context menu. Repeat this process or select a different control when you want to turn off this option.

Hiding the Toolbox
By default, the Toolbox is always visible when a design is open. If you prefer to hide it when it is not needed, click the push pin icon in the top right corner. The Toolbox will then slide out of sight when it is not in use.

To display the Toolbox when it is hidden, just move the cursor over the Toolbox label in the border. The Toolbox slides back out and remains visible as long as the cursor is over it. The push pin now appears in a horizontal position; click it to pin the Toolbox in place and prevent it from hiding.

**See also**

Choosing controls for the UI on page 491  
Customizing the Visual Designer Toolbox on page 511  
Visual Designer Editor on page 529  
Custom user-defined controls on page 479

**Getting control-specific information**

It is beyond the scope of this online help to provide detailed documentation of the numerous controls available in the Visual Designer and their properties. You can view the vendor-supplied help for a specific control by placing an instance of the control on the Design Canvas, selecting it, and pressing F1. In addition, you can use the following techniques to get useful information about the controls:

**Toolbox tool tips**

Point the cursor at a control name in the Toolbox to display a description of the control's function in a tooltip.

**Properties view help text**

With an instance of the control selected on the Design Canvas, select any property or event in the Properties view to display a description of its function at the bottom of the window.

**Class Browser**

The OpenEdge Class Browser gives you access to extensive information about the structure and content of all ABL and .NET resources, packages or namespaces, and types on your system. This information typically includes summary information about the object. It also includes details about class membership, inheritance, and interface implementation, and gives sample ABL syntax for working with the USING and DEFINE statements.

To navigate quickly to the Class Browser entry for a given object type:

1. Place an instance of the object on the Design Canvas and select it.

2. In the field at the top of the Properties view, note the name of the class from which the object is derived. The class name is the string to the right of the colon (:) that follows the instance name. Enter this class name in the Search field in the Class Browser and click to execute the search. The result list appears in the left pane of the Class Browser; in this list, select the entry that exactly matches the class name you searched for, and view the available information in the panes to the right.

For example, if you place a single SplitContainer control on the Design Canvas, by default the control instance is named splitContainer1. When you select the control, the Properties view shows its name and class thus:

```
splitContainer1 : System.Windows.Forms.SplitContainer
```

Therefore, to search for the class, enter System.Windows.Forms.SplitContainer in the Class Browser Search field.

You may find it handy to copy the class name from the corresponding DEFINE VARIABLE statement in the ABL source code and paste it into the Class Browser search. Copy the string between AS and NO-UNDO. For the preceding example, the variable declaration is:

```
DEFINE PRIVATE VARIABLE splitContainer1 AS System.Windows.Forms.SplitContainer NO-UNDO.
```
See also
Using the Properties view on page 500
Getting information from the Class Browser on page 516

Using built-in design tools
Many controls have built-in tools that can help you use the control. All of these tools provide alternative ways to set properties, so you can achieve the same results by working exclusively with the Properties view, but the built-in tools can make the process more intuitive.

The options vary among the available controls. They include "Quick Start" wizards, designers, custom property pages, and smart tags.

Quick Start wizards
Built into several OpenEdge Ultra Controls for .NET GUI, Quick Start wizards appear when you place the control on the Design Canvas. The UltraGrid is an example of a control that offers a Quick Start wizard. If you prefer not to use the wizard, you can simply click Finish. A Preferences button on the wizard lets you specify whether you want the wizard enabled and other options.

Designers
Designers assist you in the process of setting up the control. If a designer option is available, a link to it appears at the bottom of the Properties view when the control is selected on the Design Canvas. Click this link to open the tool. The ProBindingSource is an example of a control with a built-in designer.

Custom property pages
Custom property pages present a subset of the same properties that you can access through the Properties view, but in logical groups. If custom property pages are available, a link appears at the bottom of the Properties view, and also on the context (right-click) menu. Click this link to open the custom property pages.

SmartTags
SmartTags offer a subset of the most frequently used properties for the control. If SmartTags are available, the control has a small right-arrow button at the top right corner when it is selected, as in the example shown below. Click this button to open the SmartTag tool.

You can specify on the Visual Designer Preferences page whether you want smart tags enabled. This option is enabled by default.

See also
Defining properties and appearance on page 493

Using the Undo and Redo options
You can undo any editing actions that you perform on the Design Canvas, and you can redo any actions that you have previously undone. These options are not available for changes made in the Properties view.

To undo an action, select Edit > Undo or press Ctrl+Z. To redo, select Edit > Redo or press Ctrl+Y. (In some cases involving controls that have built-in design tools, you may need to use the Undo command multiple times before you see the desired result.)
The undo/redo stack for the Design Canvas is maintained separately from the stack for other Eclipse components. To undo or redo Visual Designer actions, you must have focus on the design window.

**Note:** Only editing actions that have occurred since the most recent save operation can be undone and redone; you cannot use Undo to return to a state earlier than the latest saved version of the file. To revert to an earlier saved version, right-click the class file in the Project Explorer view and select Replace With > Local History from the context menu.

**See also**
Choosing controls for the UI on page 491

**Testing a UI design**

At any time during the development of a UI design, you can test it by clicking Run on the main Eclipse toolbar while editing the file in the Visual Designer. Save the file first to see the effect of your changes, but it is not necessary to explicitly compile it or to write a startup procedure.

**See also**
Developing a visual container on page 482

**Creating custom controls**

**Creating a user control**

1. Select **File > New > ABL User Control**.
2. Enter the appropriate values as described in the New ABL User Control wizard reference help. Only the Package root and Control name fields are required. (The name must begin with a letter; spaces and most non-alphanumeric characters are not allowed.)
3. After entering the desired values, click **OK**. Progress Developer Studio for OpenEdge creates a new class file with the same name that you assigned to the control and opens it in a Visual Designer editing window. The Design Canvas displays a blank container with sizing handles that you can drag to define the size and shape of the rectangular container.
Note: You must fully define the behavior and properties of the individual controls in the container at the time that you design the user control. It is not possible to change these characteristics after placing an instance of the user control on a form. You can resize the container after placing it, however.

4. Add the desired controls to the container and define their properties, data bindings (as necessary), and event logic.

5. Save the CLS file.

After defining the custom control, you must add it to the Toolbox to make it available for use in OpenEdge GUI for .NET forms.

Note: If you move the CLS file for a user control from its original disk location to a location that is not on the project PROPATH, you must add the new location to the PROPATH. Otherwise, the type is unavailable, and the Visual Designer cannot open the control or any form containing an instance of the control.

See also

Custom user-defined controls on page 479
Defining the container size on page 483
Working with UI controls on page 485
Defining properties and appearance on page 493
Linking controls to data on page 502
Coding event logic on page 510
Customizing the Visual Designer Toolbox on page 511
New ABL User Control wizard on page 524

Creating an inherited control

Demos

Creating a Data-bound Treeview as an Inherited Control (Part 1)
Creating a Data-bound Treeview as an Inherited Control (Part 2)

Note: The OpenEdge Release 10.2B demos have not been updated for OpenEdge Release 11.0. Since 10.2B, OpenEdge Architect has been renamed as Progress Developer Studio for OpenEdge. In addition, OpenEdge Release 11.0 includes some feature enhancements and other changes. Despite the differences between 10.2B and 11.0, the demos are still a useful introduction to Progress Developer Studio for OpenEdge.

An inherited control is an extension of an existing control class. For example, you might create a custom button with specific properties and behavior that you want to incorporate in multiple UI forms.

1. Select File > New > ABL Inherited Control.

2. Enter a value in the Control name field. The name must begin with a letter; spaces and most non-alphanumeric characters are not allowed.

3. In the Inherits field, specify the control class on which the new inherited control is to be based. The best way to enter this value is to click Browse and select the parent class from the list that appears. Alternatively, you can enter the value by typing.
Note: The class from which the new control inherits must be or must inherit from System.Windows.Forms.Control.

4. The remaining fields are optional, with the exception of the Package root (required). Enter appropriate values as described in the New ABL Inherited Control wizard reference help.

5. After entering the desired values, click OK. Progress Developer Studio for OpenEdge creates a new class file with the same name that you assigned to the control. A Visual Designer editing window opens.

Note: Depending on the class from which your control inherits, it may not be possible to display a visualization of the design at this stage. In that case, the Design Canvas appears with a yellow background and a message supplied by the Microsoft .NET Framework. You can still add content to the Design Canvas and use the Properties and Events tabs to define the characteristics of the inherited control, but you do not see the GUI as it will actually appear. You can also view and edit the ABL source code (right-click in the Design Canvas and select View Source, or press F9).

You might find it helpful to create a "scratch pad" form and place an instance of the parent control on it. You can use this test form to experiment with property settings and see the results. When the control appears the way you want it, you can duplicate the values in the Properties view for your actual inherited control.

6. Add the desired controls to the container and define their properties, data bindings (as necessary), and event logic.

7. Save the CLS file.

After defining the custom control, you must add it to the Toolbox to make it available for use in OpenEdge GUI for .NET forms.

Note: If you move the CLS file for an inherited control from its original disk location to a location that is not on the project PROPATH, you must add the new location to the PROPATH. Otherwise, the type is unavailable, and the Visual Designer cannot open the control or any form containing an instance of the control.

See also
- Custom user-defined controls on page 479
- Defining properties and appearance on page 493
- Linking controls to data on page 502
- Coding event logic on page 510
- Customizing the Visual Designer Toolbox on page 511
- New ABL Inherited Control wizard on page 526

Reloading the Design Canvas while using custom controls

In the process of designing an inherited control, a user control, or a form that contains such custom controls, you might modify the contents or properties of the controls as you work. Depending on whether the element you modify is more than one hierarchical level removed from the base container, you might not see the changes on the base container until you save and reload the design. For convenience, Progress Developer Studio for OpenEdge provides a Reload Design Canvas command, accessible from the Design menu or the Visual Designer context menu.

To make sure that changes to hierarchically nested elements appear on the base container, use the following procedure:
1. Save the base container.
2. Make the desired changes to nested elements, directly editing their classes.
3. With focus on the design of the base container, select Design > Reload Design Canvas or right-click and select Reload Design Canvas. You are prompted to save the container class if you have not done so.

**See also**
- Custom user-defined controls on page 479
- Creating a user control on page 488
- Creating an inherited control on page 489

**Choosing controls for the UI**

This section explains how to populate a visual container with controls.

**Adding controls to a container**

There are several ways to place a control on a form on the **Design Canvas**:

- **Clicking** - Select a control in the **Toolbox** and click once on the form at the desired position to place an instance of the control at its default size.

- **Double-clicking** - Double-click a control in the **Toolbox** to add an instance of the control at its default size. The new control overlaps the last control that you operated on, if applicable, or appears at the top left corner of the form.

- **Dragging** - Click a control in the **Toolbox**, hold down the primary mouse button, and drag the control to the desired position on the form. When you release the mouse button, an instance of the control appears at its default size. (Alternatively, you can drag the control to a container node in the Outline view.)

- **Drawing** - Select a control in the **Toolbox**, point to the desired position on the form, and draw a rectangular outline of the size and shape you want the control to have. When you release the mouse button, an instance of the control appears, occupying the outlined area. (Some controls have shape constraints that prevent them from taking both dimensions of the outlined rectangle.)

- **Copying/cutting and pasting** - Click the pointer tool in the **Toolbox** to activate selection mode. Then select one or more controls on the **Design Canvas**, copy or cut them, and paste new instances onto the form. The Copy (Ctrl+C), Cut (Ctrl+X), and Paste (Ctrl+V) commands are available on the Edit menu and on the Design Canvas context (right-click) menu. You can also copy a control by pressing Ctrl while dragging to the desired location for the duplicate instance. Copy and paste operations are valid between separate UI classes.

  **Note:** When you add certain controls (for example, the **UltraGrid**), a Quick Start wizard may be launched.

  **Note:** You cannot add a control to a form if the form's Language property is set to any value other than **Default**. Attempting to do so results in an error message.

**Placing controls inside a container control**
In general, when you place a control on the main form, the control becomes a child object of the form. Certain controls (for example, the Panel control), however, can contain other controls; in such cases, the container control is the parent of the child objects placed within it. Adding or moving a control within the boundaries of a container control creates a parent-child relationship between the two controls.

**Note:** You can use the Outline view to make a control the child of a different parent container, without moving the control on the Design Canvas.

---

### Adding menu, toolbar, and status bar controls to an MDI Form

The ABL MDI Form offers special design-time functionality. When you click anywhere on the default container controls—the menu strip, toolbar, or status bar—an embedded design control is launched. These design controls do not appear on the form at runtime. Rather, they serve as an easy way to add controls to those container controls.

#### Adding menu controls

When you click anywhere on the menu strip, one or more text boxes with drop-down arrows appear that allow you to enter new menus and menu options. If you click a blank area on the menu strip, a single text box appears on the menu strip; typing in that box creates a new top-level menu. If you click an existing menu title, additional boxes appear within the menu structure; typing in them creates new menu options or sub-menus.

![MDI Form screenshot](image)

In addition to standard menu items, you can add ComboBox, Separator, and TextBox controls. Point the cursor inside the text box to display the down arrow to the right. Click this arrow and select the desired control from the drop-down list.

#### Adding toolbar controls

When you click anywhere on the toolbar buttons, an Add Control button \(\text{Add Control} \) appears. To use this tool to add a standard button, which is the default, just click the button. A new button with a default image appears.

The Add Control tool can also add several other control types: Label, SplitButton, DropDownListButton, Separator, ComboBox, TextBox, and ProgressBar. To add one of these controls, click the down arrow to the right of the image and select the desired control from the drop-down list.

#### Adding status bar controls

The status bar offers a similar Add Control button \(\text{Add Control} \). Click the button to add the default control, a label. Click the down arrow to select one of the other control types: ProgressBar, DropDownListButton, or SplitButton.

---

### Controls on the Design Canvas

Keep in mind that with the exception of the MDI controls described in the preceding section, the controls shown on the Design Canvas are not "live"; they are merely a representation of how the visual container will appear at runtime.

**See also**

- [Deleting controls from a container](#)
- [Using the Undo and Redo options](#)
Deleting controls from a container

To delete one or more objects from the form, select them either on the Design Canvas or in the Outline view and press Delete. Alternatively, right-click and select Delete from the context menu.

When a control is deleted from the Design Canvas:

- If it is a container control, any child controls that it contains are also deleted.
- All event-handling code related to a deleted control is removed from the class file, provided it is not used by any other control.

The Undo option is available to reverse a deletion if necessary.

When you delete a control with the Visual Designer, the assembly reference is not automatically removed from the assemblies.xml file or from the Project Explorer view. Although unused assembly references do no harm, if you want to delete them, you must do so manually. You can use the Assemblies dialog in project properties (Project > Properties > Progress OpenEdge > Assemblies). Directly editing the assemblies.xml file is not recommended.

Note: You cannot delete the visual container object (the form itself).

See also
Adding controls to a container on page 491
Selecting objects on the Design Canvas on page 493
Using the Outline view on page 494
Using the Undo and Redo options on page 487
Deleting a container on page 484

Defining properties and appearance

This section explains the techniques available for editing the appearance and behavior of UI objects that have been placed on the Design Canvas.

Selecting objects on the Design Canvas

When you want to size or move an object, edit its properties, or delete it, you must select it on the Design Canvas. You can use the following selection techniques:

- **To select a single object** - Click on the object. Alternatively, select the name of the object from the drop-down list in the field at the top of the Properties view.
- **To select multiple objects by clicking** - Shift+Click or Ctrl+Click each object that you want to select.
- **To select multiple objects by drawing** - Hold down the primary mouse button and draw a marquee on the Design Canvas around the objects that you want to select. Then release the mouse button.
- **To select all objects** - Select Edit > Select All, or press Ctrl+A.

As an alternative to making selections directly on the Design Canvas, you can use the Outline view. The Visual Designer synchronizes the selection state between the Outline view and the Design Canvas.
Selected objects appear with a visible border that includes, in most cases, selection handles at the corners and the middle of each edge. You can drag a selection handle to resize the object. (Not all controls can be resized, and some allow changes to only one dimension; selection handles appear only where they are usable.)

Note: When multiple objects are selected, certain editing operations are not available. Only actions that can be simultaneously applied to all of the selected objects are possible. Similarly, the Properties view displays only those properties and events that are common to all selected controls.

Selecting the parent object

You can select the parent object of a control via the context (right-click) menu. For example, for a toolbar button, you can select the toolstrip control that contains it.

When you right-click a control on the form, the context menu has a Select option that lists all parent objects of that control. The first-level parent is always the form itself. If the form is the only parent of the control, the form name is shown on the primary context menu (for example Select Form_1). If there are one or more lower-level parents, the Select option has a sub-menu that lists all parent objects. Use the context menu to choose the one that you want to select.

This selection technique is useful when a parent object is densely populated with child controls, as it can be difficult or impossible to click the parent on the Design Canvas. For example, a panel control might fill the form, making it impossible to click the form object. A complex design with a multi-level hierarchy of nested controls can make it difficult to select the controls visually.

Another good method for selecting parent objects is to use the Outline view.

Deselecting objects

• To deselect a single object - Shift+Click or Ctrl+Click on the object.
• To deselect all currently selected objects - Click on an unselected object.

See also
Sizing objects on page 495
Sizing the container on page 483
Positioning objects on a form on page 497
Setting tab order on page 500
Using the Properties view on page 500
Using the Outline view on page 494

Using the Outline view

The Outline view of a file that is open in the Visual Designer shows the contents of the Design Canvas and provides an alternative means of selecting, copying, deleting, renaming, and re-parenting controls. To display the Outline view if it is not open, select Window > Show View > Outline.

The Outline view shows a collapsible-expandable hierarchy of the top-level container (form or user control) and all of its controls in a tree structure. Each visual control appears as the child of its parent container. Non-visual controls appear below the tree at the same level as the base container.

Controls appear in the Outline view in the order in which you place them on the Design Canvas.

Selecting controls

You can select a control by clicking it in the tree, and you can use Ctrl+Click and Shift+Click to select multiple controls. Selecting controls in this way is equivalent to selecting them on the Design Canvas, and vice versa; the Visual Designer synchronizes the selection state between the Outline view and the Design Canvas.
In the case of complex forms, selecting items in the Outline view may be easier than selecting them directly on the Design Canvas. A complex design with a multi-level hierarchy of nested controls can make it difficult to select the controls visually.

Adding controls

As an alternative to placing a new control directly on the Design Canvas, you can select a control in the Toolbox and drag it to a container node in the Outline view. This is a useful technique if the target container is difficult to point to on the Design Canvas.

Re-parenting a control

If the form or user control includes at least one container control, you can re-parent another control - that is, make it the child of a different container - by clicking and dragging in the Outline view. Click the child control, hold down the primary mouse button, and drag the cursor so that it points to the node for the container that is to become the parent container. Then release the mouse button.

Using the context menu commands

You can select one or more controls in the Outline view and use the context (right-click) menu to cut, copy, paste, delete, or rename the selected items. (The Rename command is unavailable if more than one control is selected.)

Copying a container control copies all of its child controls as well.

When pasting, first select the target container and point to it while you right-click. If you use the Paste command without selecting a target container, the new control becomes a child of the top-level container.

Using the Undo and Redo commands

The Undo and Redo commands on the Edit menu are not available when the Outline view has focus. However, you can switch focus to the Design Canvas and then undo or redo a command that you executed in the Outline view.

See also

Selecting objects on the Design Canvas on page 493
Positioning objects on a form on page 497
Using the Undo and Redo options on page 487

Sizing objects

You can change the dimensions of objects on the Design Canvas in any of the following ways:

- Dragging selection handles
- Using shifted arrow keys to change dimensions in one-pixel increments
- Explicitly setting the values of the Size property
- Duplicating the dimensions of a specified base object
- Sizing objects to grid dimensions

Note: Certain controls have inherent constraints that prevent one or both of their dimensions from being changed (for example, you cannot change the height of a combo box control; a toolbar button cannot be resized). If you specify a new value for a protected dimension, the change is ignored.

Sizing objects by dragging

To size objects by dragging, select one or more objects that you want to operate on. Click on a selection handle on one of the objects and hold down the primary mouse button while you drag that object to the desired shape. Changes to the dimensions of the dragged object are applied in equal increments to all selected objects.
Once you have sized controls the way you want them, you can lock them to protect against their being resized by dragging. To do so, right-click on the Design Canvas and select Lock Controls. A locked control shows a small padlock icon at its top left corner when selected. This option does not prevent resizing by setting values in the Properties grid, nor does it apply to controls that you add after you invoke the locking option. To turn off the option, select Lock Controls from the context menu again.

**Changing dimensions by increments of one pixel**

To increase or decrease width or height by increments of a single pixel, select one or more objects that you want to operate on. Hold down Shift and press an arrow key as follows:

- **Shift + right arrow** - To increase the width by one pixel
- **Shift + left arrow** - To decrease the width by one pixel
- **Shift + down arrow** - To increase the height by one pixel
- **Shift + up arrow** - To decrease the height by one pixel

**Sizing objects by setting property values**

To size objects explicitly to precise dimensions, select one or more objects that you want to operate on. In the Properties view, click the Properties tab and find the Size property in the grid.

- **When a single object is selected** - The Size row of the grid shows the current dimensions (in pixels) in the right (value) column as two comma-separated values, width first followed by height. You can expand this row by clicking the plus symbol (+) to show separate Width and Height properties. You can edit the values in place in any of these three rows.

- **When multiple objects are selected** - The Size row of the grid is blank and is not expandable. However, you can enter the desired dimensions (in pixels) in the right column as two comma-separated values, width first followed by height. The dimensions you specify are applied to all selected objects.

**Sizing objects by duplicating dimensions**

Use this technique when you want to make one or more objects the same size (height, width, or both) as another existing object. Select the base object, that is, the object whose dimension(s) you want to duplicate, select the target objects to which you want to apply the dimension(s) of the base object. Select Design > Size. From the sub-menu, select Width, Height, or Both to apply one or both dimensions of the base object to the target objects.

**Sizing objects to grid dimensions**

The Design Canvas has an underlying grid (not visible on the form at run time) that is available to help you size and position objects. You can set various options related to use of the grid on the Visual Designer Preferences page. These options include the horizontal and vertical spacing of the grid lines.

The Size To Grid option is a quick way to adjust the size of selected objects so that their dimensions are multiples of the specified grid spacing values. The adjustment either enlarges or shrinks each object to the nearest exact multiple. For example, suppose you are using the default grid spacing of eight pixels for both horizontal and vertical dimensions. If you apply the Size To Grid command to a control that measures 17 pixels wide by 23 pixels high, the control is resized to 16 pixels wide by 24 pixels high (the nearest exact multiples of 8 to 17 and 23, respectively).

To use this command, select the controls to be resized and then select Design > Size > Size To Grid. Note that this command also snaps the objects to the nearest grid lines.

**See also**

- Selecting objects on the Design Canvas on page 493
- Using the Properties view on page 500
Positioning objects on a container

The Visual Designer provides several methods of positioning controls relative to the form or container control and to one another. These methods include:

- Directly moving objects by dragging or by using the arrow keys
- Spacing and aligning objects relative to one another or to the grid
- Ordering layers of objects
- Docking

See also

Selecting objects on the Design Canvas on page 493
Using the Outline view on page 494
Sizing objects on page 495
Sizing the container on page 483

Directly moving objects by dragging or by using the arrow keys

You can move a selected object or set of objects either by dragging with the mouse or by using the arrow keys.

Visual Designer preference settings

The precise way the direct-moving techniques work depends on your Visual Designer preference settings. You work in one of two layout modes, Snap to grid (the default setting) or Snap lines:

- Snap-to-grid mode - The grid underlying the Design Canvas is activated, and grid-related options become available. These options help you achieve consistent spacing and alignment of controls. If you enable the Snap To Grid option, the top left corner of any control that you place or move on the form automatically aligns to the nearest intersection of horizontal and vertical grid lines. See the reference topic on the Visual Designer Preferences page for details about these options.

- Snap-lines mode- The grid is inactive (although its height and width values still have meaning for some spacing commands). When you place or move an object on the form, the object snaps into alignment with the nearest control, and horizontal or vertical lines appear to highlight the aligned reference points (for example, the top edges, the right edges, or the text baselines). You can override the automatic alignment, if you wish, by simply moving the object.

Dragging

To move one or more objects with the mouse, select them and then move the cursor over one of the selected objects. When the cursor appears as a four-way arrow, press and hold the primary mouse button, drag to the desired position, and release the mouse button. If the Snap to grid option is enabled, placement of the objects is constrained accordingly. If you are working in snap-lines mode, you may see horizontal or vertical alignment lines as the objects you are moving approach other objects on the form.

Moving with arrow keys

You can also select one or more objects and then use the arrow keys to move them up, down, left, or right. If the Snap To Grid option is enabled, each time you press an arrow key, the objects move to the next grid alignment position. Otherwise, each arrow-key press moves the objects one pixel.

Locking control positions

Once you have positioned controls the way you want them, you can lock them to protect against their being moved. To do so, right-click on the Design Canvas and select Lock Controls. (This option does not apply to controls that you add after you invoke the locking option.) A locked control shows a small padlock icon at its top left corner when selected. To turn off the option, select Lock Controls from the context menu again.
See also
Spacing objects on page 498
Aligning objects on page 498
Ordering layers of objects on page 499
Docking objects on page 499

Spacing objects

The Design menu provides commands that facilitate regular spacing and alignment of controls. Use this procedure to regulate spacing of objects relative to one another. All space calculations are relative to horizontal or vertical edges of the affected objects.

1. Select two or more objects between which you want to adjust spacing. In most cases, the first object that you select - the base object - will not move; the objects that you select subsequently will move in response to the commands you choose. (An exception to this rule may occur with the Make Equal command.)

2. Select Format > Horizontal Spacing or Format > Vertical Spacing, and choose the appropriate command:

- **Make Equal** - The outermost objects remain in their original position, while any objects between those objects move so that all are equally spaced.
- **Increase** - Space equal to the grid width or grid height value specified in your preferences is added to the existing space between each pair of objects.
- **Decrease** - Space equal to the grid width or grid height value specified in your preferences is removed from the existing space between each pair of objects.
- **Remove** - All horizontal or vertical space between each pair of objects is removed, so that an edge of each object is horizontally or vertically adjacent to an edge of the next.

See also
Directly moving objects by dragging or by using the arrow keys on page 497
Aligning objects on page 498
Ordering layers of objects on page 499
Docking objects on page 499

Aligning objects

To align objects relative to one another:

1. Select two or more objects that you want to align. The first object that you select - the base object - will not move; the objects that you select subsequently will move in response to the commands you choose.

2. Select Format > Align and choose the desired alignment method. You can align the left, right, top, or bottom edges, or the horizontal or vertical centers.

To center one or more objects horizontally or vertically on the form, select Format > Center In Form and choose the desired option.

To align objects to the grid, select Format > Align > To Grid. The top left corner of each object snaps to the nearest intersection of horizontal and vertical grid lines.

See also
Directly moving objects by dragging or by using the arrow keys on page 497
Spacing objects on page 498
Ordering layers of objects on page 499
Docking objects on page 499
Ordering layers of objects

Ordering refers to the relative position of the virtual layers that are created when the boundaries of two or more objects overlap. To make one object appear to be in front of another, you move it forward. To make it appear to be behind the overlapping object, you move it back.

To order objects:

1. Select one or more objects on the form.
2. Select Design > Order and choose Send to Back or Bring to Front.

See also

Docking objects on page 499
Aligning objects on page 498
Spacing objects on page 498
Directly moving objects by dragging or by using the arrow keys on page 497

Docking objects

Docking locks an object in place along the full length of one edge of its parent object, or causes the object to fill the entire area of the parent object. You can set an object's docking state (the default is None) in the Properties view. Docking an object disables the ability to move or align it. You can resize a docked object only by moving the one edge that is not adjacent to an edge of the parent object.

To dock a child object in a parent object:

1. Add the parent object (for example, a Panel control). Skip this step if you are docking a control to the form itself.
2. Place the child object inside the parent object.
3. Select the child object.
4. In the Properties view, click the Properties tab and find the Dock property in the grid.
5. Click in the right column of the row and then click the down arrow that appears at the right edge of the cell. A rectangular diagram appears with buttons representing the possible docking positions:

![Docking Options Diagram]

6. Click the button corresponding to the docking option you want. If you choose the large button in the center, the docked object will completely fill the parent object.

See also

Ordering layers of objects on page 499
Aligning objects on page 498
Spacing objects on page 498
Directly moving objects by dragging or by using the arrow keys on page 497
Setting tab order

You can specify the sequence in which controls on the form receive focus when the user presses the Tab key.

**Note:** Most controls have both an AcceptsFocus property and a TabStop property, both of which must be set to True (as is generally the case by default) to make the control part of the tab sequence. If either of these properties is set to False, the control is skipped in the sequence. Toolbars and menu strips items have a default TabStop property setting of False; the buttons, menu items, and other controls contained in a toolbar or menu strip control have no TabStop property.

To set the tab order:

1. Select **Format > Tab Order** to activate tab order mode. An index number appears in the top left corner of each control that has a TabStop property. In this mode, you cannot add, select, or move objects.

   **Note:** The index number for a control parented to the form is an integer from 0 to n, where n is one less than the total number of such child controls the form. Controls parented to container controls have a decimal index number that begins with the index of the parent control. The default index numbers correspond to the order in which the controls were added to the design.

2. To change the tab order, change index numbers by clicking on the controls. Each click increments the rightmost decimal place of the control's index by 1, or resets it to 0 when the end of the sequence is reached. The system does not validate against duplicate index numbers. In the case of duplicates, precedence is determined by the relative top-to-bottom or left-to-right positions of the controls.

You may want to run the control (click **Run** on the Workbench toolbar) to test the assigned tab order.

To deactivate tab order mode, press **Esc** or select **Format > Tab Order** again.

**See also**

- Selecting objects on the Design Canvas on page 493

Using the Properties view

The Properties view displays all properties defined for the selected UI object and enables you to edit their values. On a separate tab, it also displays all events and helps you quickly create subscriptions and event methods for the ones you want the application to respond to.

**Note:** When multiple objects are selected, the Properties view displays only those properties and events that are common to all of the objects and can be set simultaneously for all of them. Be careful when editing multiple objects simultaneously, making sure that the values you set are appropriate for all of the objects.

**See also**

- Selecting objects on the Design Canvas on page 493
- Coding event logic on page 510
- Visual Designer Properties view on page 530

Viewing properties and events in the Properties view
The field at the top of the **Properties** view identifies the selected object by its name and the name of the class of which it is an instance. In the case of an inherited object, the name of the parent class appears as a prefix. This field is blank if multiple objects are selected. Properties for the selected object are displayed on the **Properties** tab in a two-column grid. Each row is a name-value pair, with the name of the property on the left and its current value on the right.

Similarly, events are displayed on the **Events** tab in a two-column grid. On this tab, the left column shows the supported events for the selected object, and the right column shows the event method, if any, that is triggered when the event is raised.

When you select a property or event in the grid, by default, a brief description of its purpose appears in a text area at the bottom of the **Properties** view. If you prefer to suppress this feature, right-click anywhere in the view below the toolbar and select **Description** from the context menu. You can toggle the option on again in the same way.

**See also**
- Setting Property values in the Properties view on page 501
- Sorting the property and event list in the Properties view on page 501

**Sorting the property and event list in the Properties view**

You can sort both the property list and the event list either alphabetically or by categories.

To toggle from one setting to the other, right-click in the view and select the sort option from the context menu.

You can also use the buttons on the **Property** view toolbar. Click **Sort by Categories** or **Sort Alphabetically**.

**See also**
- Setting Property values in the Properties view on page 501
- Viewing properties and events in the Properties view on page 500

**Setting Property values in the Properties view**

---

**Note:** This topic discusses only the **Properties** tab of the **Properties** view. Refer to **Coding event logic** for instructions pertaining to event handling.

---

You edit property values directly in the right column of the grid. The technique for specifying a value depends on the property:

- **Text entry** - Many properties take a user-defined string or integer as a value. To enter a value, you simply click in the cell and type.

- **Selection tool** - Other properties let you use either a drop-down list or a custom editor or browser to select valid options. In these cases, when you click in the value cell, a button with a down arrow or an ellipsis appears at the right. Click the button to launch the selection tool.

Depending on the control, one or more custom designers may also be available. If so, by default, the text area at the bottom of the **Properties** view includes links that you can click to launch these tools. If you prefer to suppress these links, right-click anywhere in the view below the toolbar and select **Commands** from the context menu. You can toggle the display of the links on again in the same way.
Values are saved and take effect immediately. Any values that you enter (changing the default setting) appear in bold print, making it easy to see where you have made changes.

**Note:** If you set the Language property of a form to any value other than Default, you cannot add controls to the form.

**Undoing property value changes**

The **Undo** and **Redo** options are not available in the **Properties** view. However, after making changes, you can reset properties to their default values. To do so, select the property that you want to reset and click **Reset** on the **Properties** view toolbar.

**See also**

- Sorting the property and event list in the Properties view on page 501
- Viewing properties and events in the Properties view on page 500

**Linking controls to data**

Various UI controls are capable of displaying data retrieved from a database at run time. To enable your OpenEdge GUI for .NET to display such data, you add both a visual data-displaying control and a non-visual binding source control to the visual container. You write application code that defines an ABL data source (a query, temp table, or data set) and sets the handle property of the binding source to the query handle.

**Creating a binding source object**

The binding source object, ProBindingSource, is a non-visual control whose properties define the schema for the data to be displayed by the accompanying visual control. The ProBindingSource control includes a designer tool that helps you define this schema. The tool also gives you the option of importing the schema from an XML schema (XSD) file, or from an ABL source file (like p, cls, w, i, and html).

**Note:** When designing data-bound controls, it is a good idea to establish a SQL connection to the database, so that you can see its schema in the DB Structure view. You can simplify the design process by dragging and dropping schema elements from the DB Structure view. If you choose to define the schema manually, you can refer to the DB Structure view to determine the correct table names, field names, and character types.

Begin by connecting your project to the database via an ABL connection.

**See also**

- Data-bound controls on page 478
- Editing a ProBindingSource control on page 505
- Defining a recursive relationship between tables on page 508
- Binding to an ABL data source on page 506
- Data binding example on page 506
- ProBindingSource Designer on page 527
- Using the Available Schema pane on page 529

**Drag-and-drop schema definition**

The ProBindingSource Designer supports the following drag-and-drop operations:

- You can drag table and column names from the DB Structure view to a form on the **Design Canvas**.
You can drag an ABL source file containing a schema definition (for example, a ProDataSet or a temp table) from the Project Explorer view to a form on the Visual Designer **Design Canvas**.

Both of these operations create a new binding source object, which you can then edit if necessary.

In both cases, you can drop the selected schema elements either on a blank area of the form, or on an instance of a data-displaying control. Dropping onto a control automatically sets the new binding source object as the value of the control's DataSource property.

**Note:** Drag-and-drop operations from the DB Structure view or the Resources view are not supported while the ProBindingSource Designer is open. This is an inherent limitation of modal dialogs.

Follow these steps:

1. Create or edit the container (ABL Form, ABL Dialog, or ABL MDI Form). Optionally, add the control that is to display the data. (You can add the control later if you prefer.)

2. Select the tables in the **DB Structure** view, or select the ABL source file in the **Resources** view.

   When making selections in the DB Structure view, note the following:
   - Selecting a table also selects all of its columns.
   - Use CTRL+click to select multiple tables.
   - You can select individual or multiple columns from the same table or different tables.

3. Drag the selection to the Design Canvas and drop it on the container. If the target control already exists, drop the selection on the control. The ProBindingSource Designer appears and displays the selected schema elements in the **Available schema** pane.

   If the **Project Explorer** view **Link with Editor** option is enabled, it might interfere with dragging a currently open source file onto the Design Canvas, because selecting the file in the **Project Explorer** view shifts focus to the editor buffer for the selected file. To avoid this problem, turn off the **Link with Editor** option, split the editing area so that it simultaneously shows both files, or close the source file.

4. In the **Available schema** pane, select the tables and columns that you want to add to the current schema definition. When selecting the tables and columns, note the following:
   - Selecting a table also selects all of its columns.
   - You can select multiple tables. All the selected tables are added to a default top-level table called **Table** in the Tables pane.
   - You can select individual columns, either from a single table or multiple tables. In both cases, the selected columns are added to a default top-level table called **Table**.
   - You can partially select tables and its columns. The selected schema elements are added to a default top-level table called **Table** in the Tables pane.

5. Click **Add**. The selected tables appear in the **Tables** pane and the columns appear in the **Fields** pane.

6. If necessary, edit the schema by using any of the techniques explained in **Manually defining the schema** on page 504, starting with Step 4 on page 504.

7. Click **OK** to save the current schema definition.
See also
- Importing a schema on page 504
- Manually defining the schema on page 504

Manually defining the schema
To create a binding source object and define the schema manually:

1. Create or edit the container (ABL Form, ABL Dialog, or ABL MDI Form).
2. Select the ProBindingSource control in the Toolbox and place an instance of it on the form. An object with the default name bindingSource1 appears in the non-visual control tray at the bottom of the Design Canvas, and the ProBindingSource Designer appears automatically.

3. Click Add Table. A top-level table node appears in the Tables pane, with one default field node beneath it.
4. Use the toolbar buttons to add, remove, or reorder elements, or drag and drop to change the table hierarchy:
   - To add a table, select the existing table that is to be the parent of the new one, and click .
   - To add a field, select the existing table that is to contain the field, and click .
   - To remove a schema element, select it and click . Deleting a table deletes all of its children.
   - To change the order of elements, select one or more tables or fields and click or to move them up or down. You cannot use this technique to change the hierarchical relationship of elements; you can move them only within their current level. To move multiple tables, you must select only sibling elements (that is, tables that have the same immediate parent).
   - To change the table hierarchy, select one or more sibling tables and drag them, pointing the arrow cursor at the table that is to become their parent. Release the mouse button to complete the operation. Dragging columns is not supported.

5. To import the schema, see Importing a schema on page 504
6. You can edit the name of a table node, and the name, label, and data type of a field node. To do so, select the element in the Tables or Fields pane, and change the current values as appropriate by typing directly in the rightmost pane.

7. Click OK to save the current schema definition.

See also
- Importing a schema on page 504
- Drag-and-drop schema definition on page 502

Importing a schema
You can import a schema definition from any of the following sources:

- A database connected to the current project
- A valid XSD file generated by means of the ABL function WRITE-XMLSCHEMA
- An ABL source file containing a schema definition, such as a ProDataSet

You have the option of either adding the imported schema to the current schema definition, or replacing the current definition with the imported one.

To import a schema definition:
1. Open the ProBindingSource Designer. You can do this by drag-and-drop or by using the Toolbox, as described in the two preceding sections; or you can edit an existing binding source object.

2. If you want to add an imported schema to the current schema definition, in the tree view in the Tables pane, select the table that is to be the parent of the imported schema.

3. Click to import from a connected database, or click to import from a file.
   - If importing from a database, select the desired schema elements.
   - If importing from a source file, browse to and select the file.


5. In the Available Schema pane, select tables and columns that you want to add to the current schema definition.

6. Click Add. The selected tables appear in the Tables pane and the columns in the Fields pane. You can define a recursive relation between tables in the current schema definition.

7. The schema defined using the ProBindingSource Designer appears as a ProBindingSource object in the Design Canvas.

See also
Manually defining the schema on page 504
Drag-and-drop schema definition on page 502

Editing a ProBindingSource control
You can use the ProBindingSource Designer to edit the schema and schema properties of an existing binding source object:

1. Open the ProBindingSource Designer by doing one of the following:
   - Select the ProBindingSource control on the Design Canvas, click and then ProBindingSource Designer.
   - Double-click ProBindingSource Designer on the Design Canvas.

   The ProBindingSource Designer appears and displays the current schema definition.

2. Add or remove schema elements, or modify their properties, as explained in Manually defining the schema.

3. To save the schema definition, click OK.

Note: You cannot use drag-and-drop techniques to add schema elements to an existing ProBindingSource control from the Resources view or the DB Structure view.

See also
Data-bound controls on page 478
Creating a binding source object on page 502
Binding to an ABL data source on page 506
Defining recursive relationship between tables on page 508
Data binding example on page 506
ProBindingSource Designer on page 527
Binding to an ABL data source

To enable the ProBindingSource object to pass data to the data-displaying control on the UI form, you must write application code (such as a query) that defines an ABL data source for the and sets the handle property of the binding source to the data source handle. You add this code to the form class.

The data-binding example in the next topic includes an example of such a query.

See also
Data-bound controls on page 478
Creating a binding source object on page 502
Editing a ProBindingSource control on page 505
Data binding example on page 506
ProBindingSource Designer on page 527
Data binding example on page 506

Data binding example

This topic provides step-by-step instructions for creating a data-displaying control and binding it via a ProBindingSource object to an ABL query. The exercise assumes you want your UI to include a data grid that displays the name, city, and state of all customers in the sports2000 database who live in Texas.

To begin, make sure that you have you have configured both an ABL connection and a SQL connection to the sports2000 database, and that the database is connected to your project. Also make sure that the DB Structure view is open (if not, select Window > Show View > OpenEdge Editor > DB Structure).

1. To create the data-displaying control, you must create a form and add a control capable of displaying data:
   a) Create a new ABL Form container.
   b) In the Toolbox, select either a DataGrid control or, if you have installed the OpenEdge Ultra Controls for .NET, an UltraGrid. (If you use the UltraGrid, you can just click Finish to dismiss the Quick Start wizard.)

2. To create a binding source object and associate it with the grid control:
   a) In the DB Structure view, expand the nodes of the SPORTS2000 database tree to show the columns in the Customer table.
   b) Select the Name, City, and State columns. Use CTRL+Click to select multiple items.
   c) Drag the selection from the DB Structure view and drop it on the grid control on the form. (By doing this, you automatically associate the binding source object with the grid.) The ProBindingSource Designer appears with the Customer table and the three columns that you selected in the Available schema pane on the left.
   d) In the Available schema pane, select the Customer node check box.
   e) Click Add. The Customer table that you selected appears in the Tables pane and its three columns appear in the Fields pane.
   f) To define a recursive relationship for the Customer table, select a table name from the Recursive table list box in the Properties pane. The Customer table is called the base table. Recursive table lists the name of base table and all its parent table in a hierarchical order.
   g) Click OK to save the current schema definition. The columns labeled Name, City, and State appear in the DataGrid or UltraGrid control on the form.
3. To bind to the ABL data source, define a query to retrieve the records for all customers who live in Texas:
   a) Open the class file for the form in the OpenEdge ABL Editor by right-clicking on the form and selecting View Source.
   b) Define handle variable qh, define the query, and set the binding source handle to the query handle. Here is an example of how your code might look:

   ```abl
   DEFINE VAR qh AS HANDLE NO-UNDO.
   CONSTRUCTOR PUBLIC Form1 ( ):
      SUPER().
      InitializeComponent ( ).
      CREATE QUERY qh.
      qh:QUERY-PREPARE("FOR EACH customer where customer.state = 'TX'").
      qh:QUERY-OPEN ().
      proBindingSource1:handle= qh.
   END CONSTRUCTOR.

   DESTRUCTOR Form1 ( ):
      IF VALID-HANDLE(qh) THEN DO:
         qh:QUERY-CLOSE().
         DELETE OBJECT qh.
      END.
   END DESTRUCTOR.
   
   c) Save the class file.

4. To test the data-bound grid control you designed, open the form in the Visual Designer and click Run on the main Eclipse toolbar. The control should display the retrieved records from the database:

   ![Image of data-bound grid control]

   - Drag a column header here to group by that column.
   - Thundering Surf Inc. TX
   - Wheels-a-mania TX
   - Sports Fever TX
   - Athletic X-press TX
   - Hub City Sports TX
   - Into The Outdoors Inc TX
   - Arlington Sporting Goods TX
   - Coffee City
   - Jeacott
   - Amarillo
   - Amarillo
   - Alice
   - Abilene
   - Arlington

Note: You can open the ProBindingSource Designer again at any time to edit the schema. To do so, select the object on the Design Canvas and either click and then ProBindingSource Designer, or click ProBindingSource Designer at the bottom of the Properties view.
Defining recursive relation

The ProBindingSource Designer dialog allows you to define recursive relationships between tables using the Recursive table property.

When using the Recursive table property, note of the following:

- **Recursive table** lists the name of the selected table and all its parent tables in a hierarchical order.
- A table can have a recursive relation with either itself or any of the parent table in its hierarchy.
- You can define only one recursive relation in a schema definition.

Defining a recursive relation between tables

To define a recursive relation between tables:

1. Open the ProBindingSource Designer by selecting the ProBindingSource control on the Design Canvas, click and then ProBindingSource Designer or double-clicking ProBindingSource control on the Design Canvas.

   The ProBindingSourceDesigner dialog appears and displays the current schema definition.

2. In the Tables pane, select a table for which you want to define the recursive relation. The selected table is called the base table.

3. In the Properties pane, select a table from the Recursive table list box with which you want to define a recursive relation to the base table. You can do one of the following:

   - If you want to define a recursive relation with the base table itself, select the base table from the Recursive table list box.
   - If you want to define a recursive relation with any of its parent tables, select a parent table listed in a hierarchical order from the Recursive table list box.

   **Note:** Recursive table lists the name of the base table and all its parent table in a hierarchical order. By default, no tables are defined as recursive.

4. If a recursive relation already exists for the selected schema definition, you will receive a warning message recursive relation already exists on the <table name>. Do one of the following:
Tasks

- To retain the existing recursive relation and cancel the new recursive relation, click **No**.
- To override the existing recursive relation with the new defined recursive relation, click **Yes**.

5. To save the changes in the selected schema definition, click **OK**.

**See also**

- Renaming the recursive relation between tables on page 509
- Deleting a recursive relation between tables on page 509

Renaming the recursive relation between tables

Renaming the recursive relation defined between tables involves changing the table name that is in a recursive relation with the base table. This does not delete the recursive relation defined between these tables.

To rename a recursive relation defined between tables:

1. Open the ProBindingSource Designer by selecting the **ProBindingSource** control on the **Design Canvas**, click  and then **ProBindingSource Designer** or double-clicking **ProBindingSource control** on the **Design Canvas**.
   
   The **ProBindingSource Designer** appears and displays the current schema definition with the defined recursive definition.

2. In the **Tables** pane, select the table which is in recursive relation with the base table.
3. In the **Name** field in the **Properties** pane, type a different name for the selected table.
4. To save the changes in the selected schema definition, click **OK**.

**See also**

- Deleting a recursive relation between tables on page 509
- Defining a recursive relation between tables on page 508

Deleting a recursive relation between tables

To delete a recursive relation defined between tables:

1. Open the ProBindingSource Designer by selecting the **ProBindingSource** control on the **Design Canvas**, click  and then **ProBindingSource Designer** or double-clicking **ProBindingSource control** on the **Design Canvas**.
   
   The **ProBindingSource Designer** appears and displays the current schema definition with the defined recursive definition.

2. In the **Tables** pane, select the base table for which recursion is defined.
3. From the **Recursive table** field in the **Properties** pane, select **Default**. This resets the Recursive table property to blank and deletes the recursive relation defined between the tables.
4. To save the changes in the selected schema definition, click **OK**.

**See also**

- Renaming the recursive relation between tables on page 509
- Defining a recursive relation between tables on page 508
Coding event logic

To make the objects in the UI respond to user interaction or system events, you must subscribe to the appropriate events and write methods for them.

Creating event subscriptions

A UI control typically recognizes many events, but the application responds only to the ones it subscribes to. Most controls and forms have a default event. For example, the default for a button control is the Click event, and the default for a form object is the Load event.

- To subscribe to an object's default event - Double-click the object on the Design Canvas.

- To subscribe to any event - Select the object on the Design Canvas and double-click the event name on the Events tab in the Properties view.

When you double-click in either of these ways, the Visual Designer generates the appropriate SUBSCRIBE statement in the source code. It also generates an event-handling method. Progress Developer Studio for OpenEdge automatically opens the class file in the ABE Editor (or switches to it if it is already open) and positions the cursor at the method definition.

**Note:** If you wish, you can disable the automatic generation of event code at the Visual Designer Preferences page. Select Window > Preferences > Progress OpenEdge > Visual Designer and uncheck the Generate event handler on double-click option.

Unsubscribing to events

To delete an event subscription, simply delete the value (the method name) that appears for the event in the right column of the Events tab in the Properties view.

By default, deleting this event value removes the event-handling method, as well as the event subscription, from the ABL code. If you want to change this behavior so that the method is retained, you can do so either for all events or for specific selected events.

- To keep all event-handling methods when the corresponding subscription is deleted - Go to the Visual Designer Preferences page and clear the check box labeled Remove event handlers when no longer needed.

- To protect a specific event-handling method from deletion - Locate the method declaration in the ABL source code. Immediately preceding the method, an annotation appears: @VisualDesigner. Delete this line.

**Note:** Do not change or delete any other annotations added to the ABL code by the Visual Designer. The line preceding an event-handling method is the only Visual Designer annotation that you should modify.
Coding event methods

As described in Creating event subscriptions, the quickest way to define an event method is to subscribe to the event by double-clicking in the Design Canvas or the Events tab. When you do this, you are taken to the automatically generated event-handling method in the ABL code, which is similar to the following:

END METHOD.

Add the appropriate event-handling code inside the method definition block.

The double-click technique minimizes typing and mistakes, but you can also use the Properties view to associate any method with an event. On the Events tab, click in the value cell to the right of the event name. The down-arrow button that appears lets you select from a drop-down list of all methods defined in the source code whose signature matches that of the selected event. Alternatively, you can type a method name in the cell.

Note: If you wish, you can disable the automatic generation of event code at the Visual Designer Preferences page. Select Window > Preferences > Progress OpenEdge > Visual Designer and uncheck the Generate event handler on double-click option.

You can change the name of the event handler method by editing the value on the Events tab. The change is reflected in the source code.

See also
Event behavior on page 480
Creating event subscriptions on page 510
Visual Designer Properties view on page 530

Customizing the Visual Designer Toolbox

You can set up the Toolbox to suit the particular needs of each of your OpenEdge projects, and you can select a docking position in the Visual Designer editing area. A Toolbox configuration applies to the work you do within a single project. You can use different settings for different projects. You can share your settings for any given project with other users or copy them to other projects in your own workspace.
Sharing Toolbox settings

If you make changes to the Toolbox configuration, the Visual Designer creates a file called toolbox.xml, which specifies the new custom settings, in the project root folder. You can copy this file to other projects, or share the file with other users, to duplicate the customized Toolbox settings.

If you share a Toolbox, make sure all of the controls are installed on all machines. Controls that are specified in the toolbox.xml file but are not available on the system appear in the Toolbox with a red icon.

See also
Managing control groups on page 512
Managing controls on page 513
Protecting Toolbox customizations on page 514
Restoring default Toolbox settings on page 515
Choosing the Toolbox docking position on page 515
Visual Designer Editor on page 529

Managing control groups

As installed, the Toolbox has the following control groups:

- Microsoft Controls
- OpenEdge Controls
- Telerik UI for WinForms
- OpenEdge Ultra Controls (if installed)

You can add custom control groups. After adding them, you can rename or delete custom control groups. You cannot rename or delete the default groups installed with Progress Developer Studio for OpenEdge.

If you want to save a customized configuration for later use, back up the project's toolbox.xml file before making any changes.

- To add a control group to the Toolbox - Right-click in the Toolbox, select Add Control Group from the context menu, and enter a name for the new group in the text box. To cancel the operation, press Esc.

- To rename a control group - Right-click the control group title bar, select Rename Control Group from the context menu, and enter the new name. (You cannot rename the default control groups.)

- To delete a control group and its contents - Right-click the control group title bar, select Delete Control Group from the context menu, and click Yes at the confirmation prompt. (You cannot delete the default control groups.)

Note: Deleting controls from the Toolbox does not remove them from your system. You can add them back by restoring the Toolbox settings or by using the Add Controls command.

See also
Sharing Toolbox settings on page 512
Managing controls on page 513
Protecting Toolbox customizations on page 514
Restoring default Toolbox settings on page 515
Choosing the Toolbox docking position on page 515
Managing controls

A control group can contain any number of controls, and the same control can be included in multiple groups. You can add, rearrange, rename, and delete controls. If you want to save a customized Toolbox configuration for later use, back up the project’s toolbox.xml file before making any changes.

Unavailable controls included in Toolbox

Various conditions can result in controls that are included in the Toolbox being unavailable on your system. For example, you might add third-party controls to the Toolbox and then uninstall the controls; or, another user might give you a toolbox.xml file that includes controls you do not have.

A control that is not available appears in the Toolbox with a "ban" symbol: 🗔. You cannot place the control on a form.

See also

Sharing Toolbox settings on page 512
Managing control groups on page 512
Protecting Toolbox customizations on page 514
Restoring default Toolbox settings on page 515
Choosing the Toolbox docking position on page 515
Protecting Toolbox customizations on page 531
Visual Designer Editor on page 529

Adding controls to a control group

To add controls to the Toolbox, you need to have available on your system the assemblies that contain the controls or, for user-defined controls, the CLS files that define the controls. You cannot add a second instance of a control that is already in the Toolbox, even if multiple versions of the control are installed on your system. To include a control in more than one group, use the Copy and Paste commands.

Note: Progress Software Corporation does not support controls that are not installed by the OpenEdge installation program.

To add controls to a control group:

1. Right-click in the Toolbox and select Add Controls from the context menu. The Add Controls dialog appears.

2. The Add Controls dialog Browsed Assemblies tab is initially open. If you have browsed previously during your current Progress Developer Studio for OpenEdge session, the controls available in the last-browsed assembly appear in the list. Select one or more controls from the current list, or generate a new list by clicking Browse to locate and select one or more DLL or EXE files containing UI controls.

   Alternatively, click the Global Assemblies tab to see a list of controls available in the global assembly cache, or ABL Controls to see a list of custom user-defined controls.

   You can filter the list to display only those controls whose names begin with the characters you enter in the Filter field.

3. Check the control(s) that you want to add to the Toolbox. You can select controls on all three tabs.
4. In the field at the top of the dialog, select the control group to which you want to add the controls.

5. Click OK to add the selected controls to the Toolbox.

**Renaming controls**

To change the name of a control, right-click the control in the Toolbox, select Rename from the context menu, and enter the new name.

*Note:* Renaming a control in the Toolbox does not affect control instances previously placed on the Design Canvas, or controls of the same name in other Toolbox control groups.

**Deleting controls from a control group**

To delete one or more controls from the Toolbox, select them, using Ctrl+Click to select multiple controls. Press Delete, or right-click and select Delete from the context menu, and click OK at the confirmation prompt.

*Note:* Deleting controls from the Toolbox does not remove them from your system. You can add them back by restoring the Toolbox settings or by using the Add Controls command.

**Rearranging controls among groups**

Click and drag controls in the Toolbox to move them from one group to another, or press Ctrl as you drag to copy them to the target group. You can also use the Cut, Copy, and Paste commands, available on the context (right-click) menu.

**Disabling controls**

It is possible to disable, or "blacklist," specific controls that you do not want used. To do this, add an entry for each such control to the following file: $DLC\Properties\unSupported.xml

The entry in the unSupported.xml file must include several pieces of information about the control in a specific format. The easiest way to create a properly formatted entry is to copy it from toolbox.xml and paste it into unSupported.xml. If no such toolbox.xml file is available, generate it by customizing the Toolbox (making sure that it contains the control in question).

You must restart Progress Developer Studio for OpenEdge for the disablement to take effect. OpenEdge removes the disabled controls from all instances of toolbox.xml and prevents them from being added to any Toolbox. However, existing instances of the controls in previously created designs are not affected.

**Protecting Toolbox customizations**

To prevent changes to your current Toolbox settings, right-click in the Toolbox and select Lock Toolbox from the context menu. Repeat this process to unlock the Toolbox and allow changes.

**See also**

- Sharing Toolbox settings on page 512
- Managing control groups on page 512
- Managing controls on page 513
- Restoring default Toolbox settings on page 515
- Choosing the Toolbox docking position on page 515
- Visual Designer Editor on page 529
Restoring default Toolbox settings

You can restore the Toolbox to its default state after making changes. Doing so has no effect on control instances that have already been placed on a form.

Before restoring the Toolbox, back up the project's toolbox.xml file if you want to save the customized configuration.

To restore the Toolbox to its default state, right-click in the Toolbox and select Reset Toolbox from the context menu.

Note: The Reset Toolbox command is disabled if the project includes references to any out-of-date assembly versions. This helps prevent the possibility of a version conflict between the controls available in the Toolbox and the controls in use in the project.

See also
Sharing Toolbox settings on page 512
Managing control groups on page 512
Managing controls on page 513
Protecting Toolbox customizations on page 514
Choosing the Toolbox docking position on page 515
Visual Designer Editor on page 529

Choosing the Toolbox docking position

By default, the Toolbox is docked at the right edge of the Visual Designer editing area. You can choose a different docking position if you wish.

To specify a persistent docking position for all projects, select Window > Preferences > Progress OpenEdge > Visual Designer. Click to select the desired position. Changing the docking preference does not affect currently open Visual Designer editing windows.

To move the Toolbox only for your current editing session, position the cursor over the Toolbox heading, hold down the primary mouse button, drag to the desired docking position, and release the mouse button. Dragging the Toolbox affects only the current file, and when you close and re-open the file, the Toolbox reverts to the position specified in the Visual Designer preferences.

See also
Managing control groups on page 512
Managing controls on page 513
Protecting Toolbox customizations on page 514
Restoring default Toolbox settings on page 515
Visual Designer Editor on page 529

Working with .NET resources

This section includes the following topics on .NET-related operations outside the context of the Visual Designer.
Getting information from the Class Browser

The Class Browser allows you to view detailed information about properties, data members, constructors, methods, and events for ABL and .NET resources, packages or namespaces, and types.

The Class Browser is included by default in the Visual Designer perspective as a Fast View. To open this view, click on the Fast View toolbar. To display the Class Browser in any perspective, select Window > Show View > Other > OpenEdge Editor > Class Browser.

See also

Getting control-specific information on page 486

Adding and removing assembly references

The Visual Designer automatically adds all required assembly references to the project assemblies.xml file when you place controls on the Design Canvas. However, there may be cases where your application code calls other .NET objects that are not related to OpenEdge GUI for .NET design, or where the code was manually developed outside the Visual Designer. Progress Developer Studio for OpenEdge provides several convenient ways to manage assembly references that are not automatically maintained by the Visual Designer.

You can use any of the following procedures:

- Right-click the Referenced Assemblies node in the Project Explorer view and select Add Assembly References from the context menu. Use the Add Assembly References dialog to select assemblies and add them to the assemblies.xml file of the current project.

- Select Project > Properties > Progress OpenEdge > Assemblies. At the Assemblies property page, you can launch the Add Assembly References dialog to add references, and you can remove references that are no longer needed. These editing operations affect only the assemblies.xml file for the current project.

- In either the OpenEdge Editor perspective or the OpenEdge Visual Designer perspective, launch the external Assembly References Tool by selecting OpenEdge > Tools > Assembly References. This tool allows you to manage assembly references in any XML file on your system.

See also

.NET assemblies and the Visual Designer on page 473
Upgrading assembly references on page 516

Updating assembly references

When you install a new version of .NET controls, assembly references for projects that you developed under the previous version become out of date. If you want these projects to use the newer controls, you must update the references.

You use the Update Assembly References tool for this purpose.

You can launch the Update Assembly References tool from Progress Developer Studio for OpenEdge, but the tool actually runs outside the Eclipse framework. This topic provides basic instructions for using the tool; you can access more detailed online help when the tool is running.

When to update assembly references
You should complete the update procedure each time you install a new version of any control set. This recommendation applies whether you install a new release of OpenEdge that includes upgraded controls, a patch from Progress Software Corporation that includes upgraded controls, or a control set that you obtain from any other source. Failure to update assembly references may prevent existing OpenEdge GUI for .NET forms from running, and may also prevent the Visual Designer from opening existing forms or from functioning properly when you try to develop new forms.

Note: It is not necessary to update assemblies after applying a hot fix from Progress Software Corporation.

Automatic update of Progress assembly references

The first time you open a workspace that includes one or more projects with out-of-date assembly references, a dialog alerts you to the situation and asks whether you want to update the references. If you click Yes at this dialog, OpenEdge automatically updates all references to Progress assemblies, including those containing OpenEdge Ultra Controls (Infragistics controls).

If you decline this automatic update, you will not be prompted again to update the current workspace. You should manually run the Update Assembly References tool when you are ready to perform the update.

The automatic update procedure does not affect third-party (non-Progress) assemblies. Run the Update Assembly References tool when you install a new version of any third-party controls.

Files requiring updating

The Update Assembly References tool modifies references as needed in all instances of the files assemblies.xml and toolbox.xml. Potentially, there is one of each of these files in every project in a workspace, plus the toolbox.xml file for the global Toolbox.

Launching the Update Assembly References tool

You can launch the Update Assembly References tool either from an OpenEdge main menu or from a command prompt:

• From OpenEdge main menu, select OpenEdge > Migration > Update Assembly References.

• From a command prompt, open a Proenv command shell and enter updasmref.exe at the prompt. You can optionally include an argument specifying the assemblies.xml and toolbox.xml files to be updated as follows:

    updasmref -files file_paths

    Separate multiple file paths with commas.

Using the Update Assembly References tool

The procedure for updating the references depends on whether the controls are Progress controls (for example, OpenEdge Ultra Controls) or third-party controls. See the online help provided with the tool for instructions on each of these operations.

Updating assembly references to Progress controls

The tool automatically proposes the current version (that is, the assembly version installed with the current version of OpenEdge) for each reference to an out-of-date Progress assembly. Thus, with a single click you can update your references to Progress assemblies.

Updating assembly references to third-party controls

The tool lists all non-Progress assemblies and allows you to manually select or enter the versions that you want to use.

Restarting and recompiling

After updating assembly references, do the following:
• Restart Progress Developer Studio for OpenEdge if one or more projects in the current workspace use the updated assembly references.

• Recompile any projects that use the updated assembly references.

See also
Adding and removing assembly references on page 516
.NET assemblies and the Visual Designer on page 473

Reference

Visual Designer Preferences

The Visual Designer Preferences page allows you to set options affecting the appearance and behavior of the Toolbox and the Design Canvas. To access these settings, select Window > Preferences > Progress OpenEdge > Visual Designer.

The following options and controls are available:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Global Toolbox location</strong></td>
<td>Specifies the path of the toolbox.xml file that determines the content of the Toolbox shared by projects that use the global Toolbox option. (You set this option on the Progress OpenEdge properties page.)</td>
</tr>
<tr>
<td><strong>Show SmartTags</strong></td>
<td>Enables or disables the use of SmartTag design aids for setting frequently used control properties of various controls. If this option is enabled, controls with SmartTags have a small right-arrow button at the top right corner when selected, as in the example shown. Clicking this button opens the SmartTag tool.</td>
</tr>
<tr>
<td><strong>Generate event handler on double-click</strong></td>
<td>Enables or disables the automatic generation of an event subscription and event-handler method when a control on the Design Canvas or an event in the Properties view is double-clicked.</td>
</tr>
<tr>
<td><strong>Remove event handlers when no longer needed</strong></td>
<td>Enables or disables the automatic deletion of event-handler methods when the corresponding event subscription is deleted.</td>
</tr>
<tr>
<td><strong>Toolbox docking position</strong></td>
<td>Specifies whether the Toolbox is docked to the right, left, top, or bottom edge of the Visual Designer editing window.</td>
</tr>
<tr>
<td><strong>Snap lines or Grid</strong></td>
<td>Specifies whether controls, when placed or moved on the Design Canvas, are aligned relative to other controls or to the design grid. The Grid setting takes effect only if the Snap to grid option is enabled.</td>
</tr>
<tr>
<td><strong>Height and Width</strong></td>
<td>Specify, in pixels, the vertical and horizontal intervals of the design grid.</td>
</tr>
</tbody>
</table>
Enables or disables the visible display of the grid on forms on the Design Canvas.

Enables or disables automatic alignment to the design grid when controls are placed or moved on the Design Canvas.

Specifies the number of actions that the Visual Designer can undo or redo.

Enables or disables the inclusion of the U (untranslatable) attribute for character strings that the Visual Designer inserts in ABL code. If present, this attribute prevents the string from being processed by the OpenEdge Translation Manager.

See also
Working with UI controls on page 485
Using built-in design tools on page 487
Creating event subscriptions on page 510
Sizing objects on page 495
Positioning objects on a form on page 497

OpenEdge GUI for .NET wizards

New ABL Form wizard

The New ABL Form wizard helps you create a basic form to which you can add controls in the Visual Designer. This wizard appears when you select File > New > ABL Form.

The following controls are available:

| Package root | Specifies a currently open project to contain the class code and other project code. ClickBrowseif you want to select a project other than the current one (the default value). |
| Package | Optionally specifies a package name, corresponding to a sub-folder of the package root, to contain the class file. ClickBrowseor enter the folder path by typing. A period (.) must separate each subfolder from its parent folder in the path name. |
| Form name | (Required) Specifies the name of the class. The name must begin with a letter; spaces and most non-alphanumeric characters are not allowed. The .cls extension will be appended automatically. |
| Final | If checked, specifies that inheritance from this class is disallowed. A FINAL option is included in the generated ABL code. A final class cannot be abstract. |
| **Abstract** | If checked, specifies that the class is abstract and cannot be instantiated. An abstract class is designed to serve as a super class from which other classes inherit and implement members. Therefore, an abstract class cannot be final. |
| **Widget pool** | If checked, specifies that a USE-WIDGET-POOL option will be included in the generated ABL code. |
| **Inherits** | Optionally specifies another class in the current project as a super class from which the new class inherits state and behavior. Click Browse or enter the class path by typing. A period (.) must separate each node from its parent folder in the path name. 

**Note:** The super class cannot be generic. Moreover, even though a form can inherit from an abstract super class, the Visual Designer does not support designing such a form. To design a form that inherits from an abstract parent, you must write the code manually. |
| **Implements** | Lets you optionally specify one or more interfaces in the current project that the class implements. Click Add and select the desired interfaces at the Interface Selection dialog. Use the Remove button to remove an interface from the list after adding it. 

**Note:** The class cannot implement a generic interface. |
| **Generate default constructor** | Specifies that the class is to include a default constructor method. Because this option is required for OpenEdge GUI for .NET forms, it is automatically checked, and the control is disabled. |
| **Generate destructor** | Specifies that the class is to include a destructor method. Because this option is required for OpenEdge GUI for .NET forms, it is automatically checked, and the control is disabled. |
| **Generate super class constructors** | If checked, specifies that the class is to include all constructor methods declared in the parent class. |
| **Add routine-level error handling** | If checked, specifies that the ROUTINE-LEVEL ON ERROR UNDO, THROW statement will be included in the generated ABL code. |
| **Throw a Not Implemented exception** | If selected, specifies that "METHOD NOT IMPLEMENTED" error code will be added to stubs for members implemented via interface and stubs for inherited abstract members. |
| **Return a default value** | If selected, specifies that default return values will be added to stubs for members implemented via interface and stubs for inherited abstract members. |
Description | Lets you optionally enter a description for the class. This text will appear in the file header.

Purpose | Lets you optionally enter an explanation of the purpose of the class. This text will appear in the file header.

**See also**

[Visual container types](#) on page 475
[Developing a visual container](#) on page 482

**New ABL Dialog wizard**

The **New ABL Dialog** wizard lets you create a dialog with standard **OK** and **Cancel** buttons. An event subscription and event logic for a click are pre-coded for each of these buttons. This wizard appears when you select **File > New > ABL Dialog**.

The following controls are available:

| Package root | Specifies a currently open project to contain the class code and other project code. Click **Browse** if you want to select a project other than the current one (the default value). |
| Package | Optionally specifies a package name, corresponding to a sub-folder of the package root, to contain the class file. Click **Browse** or enter the folder path by typing. A period (.) must separate each subfolder from its parent folder in the path name. |
| Dialog name | *(Required)* Specifies the name of the class. The name must begin with a letter; spaces and most non-alphanumeric characters are not allowed. The .cls extension will be appended automatically. |
| Final | If checked, specifies that inheritance from this class is disallowed. A FINAL option is included in the generated ABL code. A final class cannot be abstract. |
| Abstract | If checked, specifies that the class is abstract and cannot be instantiated. An abstract class is designed to serve as a super class from which other classes inherit and implement members. Therefore, an abstract class cannot be final. |
| Widget pool | If checked, specifies that a USE-WIDGET-POOL option will be included in the generated ABL code. |
| Inherits | Optionally specifies another class in the current project as a super class from which the new class inherits state and behavior. Click **Browse** or enter the class path by typing. A period (.) must separate each node from its parent folder in the path name. |

**Note:** The super class cannot be generic. Moreover, even though a dialog can inherit from an abstract super class, the Visual Designer does not support designing such a dialog. To design a dialog that inherits from an abstract parent, you must write the code manually.
<table>
<thead>
<tr>
<th>Implements</th>
<th>Lets you optionally specify one or more interfaces in the current project that the class implements. Click <strong>Add</strong> and select the desired interfaces at the <strong>Interface Selection</strong> dialog. Use the <strong>Remove</strong> button to remove an interface from the list after adding it.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Note:</strong> The class cannot implement a generic interface.</td>
</tr>
<tr>
<td>Generate default constructor</td>
<td>Specifies that the class is to include a default constructor method. Because this option is required for OpenEdge GUI for .NET forms, it is automatically checked, and the control is disabled.</td>
</tr>
<tr>
<td>Generate destructor</td>
<td>Specifies that the class is to include a destructor method. Because this option is required for OpenEdge GUI for .NET forms, it is automatically checked, and the control is disabled.</td>
</tr>
<tr>
<td>Generate super class constructors</td>
<td>If checked, specifies that the class is to include all constructor methods declared in the parent class.</td>
</tr>
<tr>
<td>Add routine-level error handling</td>
<td>If checked, specifies that the ROUTINE-LEVEL ON ERROR UNDO, THROW statement will be included in the generated ABL code.</td>
</tr>
<tr>
<td>Throw a Not Implemented exception</td>
<td>If selected, specifies that &quot;METHOD NOT IMPLEMENTED&quot; error code will be added to stubs for members implemented via interface and stubs for inherited abstract members.</td>
</tr>
<tr>
<td>Return a default value</td>
<td>If selected, specifies that default return values will be added to stubs for members implemented via interface and stubs for inherited abstract members.</td>
</tr>
<tr>
<td>Description</td>
<td>Lets you optionally enter a description for the class. This text will appear in the file header.</td>
</tr>
<tr>
<td>Purpose</td>
<td>Lets you optionally enter an explanation of the purpose of the class. This text will appear in the file header.</td>
</tr>
</tbody>
</table>

**See also**

- [Visual container types](#) on page 475
- [Developing a visual container](#) on page 482

**New ABL MDI Form wizard**

The **New ABL MDI Form** wizard lets you create a form enabled for multiple-document interface functionality, meaning that it can be the parent of child forms. The form includes a menu strip and a toolbar containing common menus and command buttons with pre-coded event logic, as well as a status bar.

This wizard appears when you select **File > New > ABL MDI Form**.

The following controls are available:
<table>
<thead>
<tr>
<th><strong>Package root</strong></th>
<th>Specifies a currently open project to contain the class code and other project code. Click <strong>Browse</strong> if you want to select a project other than the current one (the default value).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Package</strong></td>
<td>Optionally specifies a package name, corresponding to a sub-folder of the package root, to contain the class file. Click <strong>Browse</strong> or enter the folder path by typing. A period (.) must separate each subfolder from its parent folder in the path name.</td>
</tr>
<tr>
<td><strong>MDI Form name</strong></td>
<td>(Required) Specifies the name of the class. The name must begin with a letter; spaces and most non-alphanumeric characters are not allowed. The .cls extension will be appended automatically.</td>
</tr>
<tr>
<td><strong>Final</strong></td>
<td>If checked, specifies that inheritance from this class is disallowed. A FINAL option is included in the generated ABL code. A final class cannot be abstract.</td>
</tr>
<tr>
<td><strong>Abstract</strong></td>
<td>If checked, specifies that the class is abstract and cannot be instantiated. An abstract class is designed to serve as a super class from which other classes inherit and implement members. Therefore, an abstract class cannot be final.</td>
</tr>
<tr>
<td><strong>Widget pool</strong></td>
<td>If checked, specifies that a USE-WIDGET-POOL option will be included in the generated ABL code.</td>
</tr>
</tbody>
</table>
| **Inherits**     | Optionally specifies another class in the current project as a super class from which the new class inherits state and behavior. Click **Browse** or enter the class path by typing. A period (.) must separate each node from its parent folder in the path name.  

**Note:** The super class cannot be generic. Moreover, even though a form can inherit from an abstract super class, the Visual Designer does not support designing such a form. To design a form that inherits from an abstract parent, you must write the code manually. |
| **Implements**   | Lets you optionally specify one or more interfaces in the current project that the class implements. Click **Add** and select the desired interfaces at the **Interface Selection** dialog. Use the **Remove** button to remove an interface from the list after adding it.  

**Note:** The class cannot implement a generic interface. |
| **Generate default constructor** | Specifies that the class is to include a default constructor method. Because this option is required for OpenEdge GUI for .NET forms, it is automatically checked, and the control is disabled. |
| **Generate destructor** | Specifies that the class is to include a destructor method. Because this option is required for OpenEdge GUI for .NET forms, it is automatically checked, and the control is disabled. |
If checked, specifies that the class is to include all constructor methods declared in the parent class.

**Add routine-level error handling**
If checked, specifies that the ROUTINE-LEVEL ON ERROR UNDO, THROW statement will be included in the generated ABL code.

**Throw a Not Implemented exception**
If selected, specifies that “METHOD NOT IMPLEMENTED” error code will be added to stubs for members implemented via interface and stubs for inherited abstract members.

**Return a default value**
If selected, specifies that default return values will be added to stubs for members implemented via interface and stubs for inherited abstract members.

**Description**
Lets you optionally enter a description for the class. This text will appear in the file header.

**Purpose**
Lets you optionally enter an explanation of the purpose of the class. This text will appear in the file header.

**See also**
- **Visual container types** on page 475
- **Developing a visual container** on page 482

**New ABL User Control wizard**
The New ABL User Control wizard lets you create a custom set of individual controls grouped in a container that you can add as a single entity to another container. This wizard appears when you select File > New > ABL User Control.

The following controls are available:

<table>
<thead>
<tr>
<th>Package root</th>
<th>Specifies a currently open project to contain the class code and other project code. Click <strong>Browse</strong> if you want to select a project other than the current one (the default value).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
<td>Optionally specifies a package name, corresponding to a sub-folder of the package root, to contain the class file. Click <strong>Browse</strong> or enter the folder path by typing. A period (.) must separate each subfolder from its parent folder in the path name.</td>
</tr>
<tr>
<td>User Control name</td>
<td>(Required) Specifies the name of the class. The name must begin with a letter; spaces and most non-alphanumeric characters are not allowed. The .cls extension will be appended automatically.</td>
</tr>
<tr>
<td>Final</td>
<td>If checked, specifies that inheritance from this class is disallowed. A FINAL option is included in the generated ABL code. A final class cannot be abstract.</td>
</tr>
<tr>
<td><strong>Abstract</strong></td>
<td>If checked, specifies that the class is abstract and cannot be instantiated. An abstract class is designed to serve as a super class from which other classes inherit and implement members. Therefore, an abstract class cannot be final.</td>
</tr>
<tr>
<td><strong>Widget pool</strong></td>
<td>If checked, specifies that a USE-WIDGET-POOL option will be included in the generated ABL code.</td>
</tr>
</tbody>
</table>
| **Inherits** | Optionally specifies another class in the current project as a super class from which the new class inherits state and behavior. Click **Browse** or enter the class path by typing. A period (.) must separate each node from its parent folder in the path name.  

**Note:** The super class cannot be generic. Moreover, even though a control can inherit from an abstract super class, the Visual Designer does not support designing such a control. To design a user control that inherits from an abstract parent, you must write the code manually. |
| **Implements** | Lets you optionally specify one or more interfaces in the current project that the class implements. Click **Add** and select the desired interfaces at the **Interface Selection** dialog. Use the **Remove** button to remove an interface from the list after adding it.  

**Note:** The class cannot implement a generic interface. |
<p>| <strong>Generate default constructor</strong> | Specifies that the class is to include a default constructor method. Because this option is required for OpenEdge GUI for .NET forms, it is automatically checked, and the control is disabled. |
| <strong>Generate destructor</strong> | Specifies that the class is to include a destructor method. Because this option is required for OpenEdge GUI for .NET forms, it is automatically checked, and the control is disabled. |
| <strong>Generate super class constructors</strong> | If checked, specifies that the class is to include all constructor methods declared in the parent class. |
| <strong>Add routine-level error handling</strong> | If checked, specifies that the ROUTINE-LEVEL ON ERROR UNDO, THROW statement will be included in the generated ABL code. |
| <strong>Throw a Not Implemented exception</strong> | If selected, specifies that &quot;METHOD NOT IMPLEMENTED&quot; error code will be added to stubs for members implemented via interface and stubs for inherited abstract members. |
| <strong>Return a default value</strong> | If selected, specifies that default return values will be added to stubs for members implemented via interface and stubs for inherited abstract members. |</p>
<table>
<thead>
<tr>
<th>Description</th>
<th>Lets you optionally enter a description for the class. This text will appear in the file header.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Lets you optionally enter an explanation of the purpose of the class. This text will appear in the file header.</td>
</tr>
</tbody>
</table>

**See also**
- [Custom user-defined controls](#) on page 479
- [Creating a user control](#) on page 488

**New ABL Inherited Control wizard**

The New ABL Inherited Control wizard lets you create a custom control class derived from an existing control. This wizard appears when you select **File > New > ABL Inherited Control**.

The following controls are available:

<table>
<thead>
<tr>
<th>Package root</th>
<th>Specifies a currently open project to contain the class code and other project code. Click <strong>Browse</strong> if you want to select a project other than the current one (the default value).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package</td>
<td>Optionally specifies a package name, corresponding to a sub-folder of the package root, to contain the class file. Click <strong>Browse</strong> or enter the folder path by typing. A period (.) must separate each subfolder from its parent folder in the path name.</td>
</tr>
<tr>
<td>Inherited Control name</td>
<td><em>(Required)</em> Specifies the name of the class. The name must begin with a letter; spaces and most non-alphanumeric characters are not allowed. The .cls extension will be appended automatically.</td>
</tr>
<tr>
<td>Final</td>
<td>If checked, specifies that inheritance from this class is disallowed. A FINAL option is included in the generated ABL code. A final class cannot be abstract.</td>
</tr>
<tr>
<td>Abstract</td>
<td>If checked, specifies that the class is abstract and cannot be instantiated. An abstract class is designed to serve as a super class from which other classes inherit and implement members. Therefore, an abstract class cannot be final.</td>
</tr>
<tr>
<td>Widget pool</td>
<td>If checked, specifies that a USE-WIDGET-POOL option will be included in the generated ABL code.</td>
</tr>
<tr>
<td>Inherits</td>
<td>Optionally specifies another class in the current project as a super class from which the new class inherits state and behavior. Click <strong>Browse</strong> or enter the class path by typing. A period (.) must separate each node from its parent folder in the path name.</td>
</tr>
</tbody>
</table>

**Note:** The super class cannot be generic. Moreover, even though a control can inherit from an abstract super class, the Visual Designer does not support designing such a control. To design a control that inherits from an abstract parent, you must write the code manually.
Lets you optionally specify one or more interfaces in the current project that the class implements. Click Add and select the desired interfaces at the Interface Selection dialog. Use the Remove button to remove an interface from the list after adding it.

**Note:** The class cannot implement a generic interface.

<table>
<thead>
<tr>
<th>Implement</th>
<th>Lets you optionally specify one or more interfaces in the current project that the class implements. Click Add and select the desired interfaces at the Interface Selection dialog. Use the Remove button to remove an interface from the list after adding it.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generate default constructor</td>
<td>Specifies that the class is to include a default constructor method. Because this option is required for OpenEdge GUI for .NET forms, it is automatically checked, and the control is disabled.</td>
</tr>
<tr>
<td>Generate destructor</td>
<td>Specifies that the class is to include a destructor method. Because this option is required for OpenEdge GUI for .NET forms, it is automatically checked, and the control is disabled.</td>
</tr>
<tr>
<td>Generate super class constructors</td>
<td>If checked, specifies that the class is to include all constructor methods declared in the parent class.</td>
</tr>
<tr>
<td>Add routine-level error handling</td>
<td>If checked, specifies that the ROUTINE-LEVEL ON ERROR UNDO, THROW statement will be included in the generated ABL code.</td>
</tr>
<tr>
<td>Throw a Not Implemented exception</td>
<td>If selected, specifies that &quot;METHOD NOT IMPLEMENTED&quot; error code will be added to stubs for members implemented via interface and stubs for inherited abstract members.</td>
</tr>
<tr>
<td>Return a default value</td>
<td>If selected, specifies that default return values will be added to stubs for members implemented via interface and stubs for inherited abstract members.</td>
</tr>
<tr>
<td>Description</td>
<td>Lets you optionally enter a description for the class. This text will appear in the file header.</td>
</tr>
<tr>
<td>Purpose</td>
<td>Lets you optionally enter an explanation of the purpose of the class. This text will appear in the file header.</td>
</tr>
</tbody>
</table>

**See also**
- Custom user-defined controls on page 479
- Creating an inherited control on page 489

**ProBindingSource Designer**

The ProBindingSource Designer lets you define the schema for data, derived from an ABL data source, to be displayed by a data-displaying control. This tool appears in the following situations:

- When you use the Toolbox to add a ProBindingSource object to the Design Canvas.
- When you select an existing ProBindingSource object and click ProBindingSource Designer on the Properties view, the context (right-click) menu, or the Smart Tag.
- After you drag a schema element from the DB Structure view or an ABL source file from the Resources view and drop it on the Design Canvas.
The following controls are available:

<table>
<thead>
<tr>
<th><strong>Available schema (left) pane</strong></th>
<th>Displays the database schema imported from a connected database, a XML Schema (XSD) file or an ABL source file (like p, cls, w, i, and html).</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Click to Hide Available Schema</strong></td>
<td>Hides the Available schema pane.</td>
</tr>
<tr>
<td><strong>Add</strong></td>
<td>Adds the table and column selected in the Available schema pane to the Tables pane and Fields pane.</td>
</tr>
<tr>
<td><strong>Tables pane</strong></td>
<td>Displays a hierarchical representation of the tables defined in the current schema.</td>
</tr>
<tr>
<td><strong>Fields pane</strong></td>
<td>Displays the fields (columns) of the currently selected table.</td>
</tr>
<tr>
<td><strong>Properties (right) pane</strong></td>
<td>Lets you define the name of each table and field, as well as each field's data type. Select a table or field in the left or middle pane to view and edit its properties. These values must match those of the actual ABL data source. You can also define field labels as you want them to appear in the GUI at runtime.</td>
</tr>
<tr>
<td><strong>Recursive Table</strong></td>
<td>This is a property for a table, used to define recursive relation between tables.</td>
</tr>
<tr>
<td><strong>Add Table</strong></td>
<td>Adds a table node. The new table is a child of the one above it in the tree.</td>
</tr>
<tr>
<td><strong>Add Field</strong></td>
<td>Adds a field node. The new field belongs to the table above it in the tree.</td>
</tr>
<tr>
<td><strong>Delete</strong></td>
<td>Deletes the selected node and its child nodes, if any.</td>
</tr>
<tr>
<td><strong>Move Up</strong></td>
<td>Moves the selected field up in the tree. This button is enabled only when a field node is selected.</td>
</tr>
<tr>
<td><strong>Move Down</strong></td>
<td>Moves the selected field down in the tree. This button is enabled only when a field node is selected.</td>
</tr>
<tr>
<td><strong>Import from Database</strong></td>
<td>Displays a Schema Selection dialog that displays the databases connected to the project and lets you select schema elements to import.</td>
</tr>
<tr>
<td><strong>Import from File</strong></td>
<td>Displays a file browser that lets you select an existing XML Schema Definition (XSD) file or ABL source file. After you select the file, a Schema Selection dialog appears and lets you select schema elements to import.</td>
</tr>
<tr>
<td><strong>Expand or Collapse the Available Schema pane.</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Add</strong></td>
<td>Adds the tables selected from Available Schema to the Tables pane.</td>
</tr>
<tr>
<td><strong>OK</strong></td>
<td>Saves the current schema definition. The defined schema appears as a ProBindingSource object in the Design Canvas.</td>
</tr>
<tr>
<td><strong>Cancel</strong></td>
<td>Closes the ProBindingSource Designer dialog without saving the changes to the schema definition.</td>
</tr>
</tbody>
</table>
Using the Available Schema pane

The Available Schema pane in the ProBindingSource Designer displays the database schema imported from a connected database, a XML Schema (XSD) file, an ABL source file, or a non ABLsource file.

| Note: | If the file is a non ABL source file, you must add the file type to the Content Types page (Preferences > General > Content Types) and the File Associations Preferences page (Preferences > General > Editor > File Associations). |

When selecting the schema elements (tables and columns) in the Available Schema pane, note of the following:

• Selecting a table also selects all of its columns.
• You can select multiple tables without selecting their columns. All the selected tables are added to a default top-level root called Table in the Tables pane.
• You can select individual columns either from a single table or from multiple tables. In both cases, the selected columns are added to a default top-level root called Table in the Tables pane.
• You can partially select the tables and its columns. The selected schema elements are added to a default top-level root called Table in the Tables pane.

See also
Data-bound controls on page 478
Creating a binding source object on page 502
Linking controls to data on page 502

Workspace components

Visual Designer Editor

The Visual Designer Editor comprises the Toolbox and the Design Canvas.

Toolbox

By default, the Toolbox is docked to the right edge of the editing area. If you wish, you can change the docking position by dragging or through the Visual Designer Preferences page.

The Toolbox contains controls organized in control groups, as shown in the following illustration:
You can customize the Toolbox by adding or removing control groups and controls.

**Control groups**

A control group is a logical grouping of individual controls. As installed, the Toolbox has the following control groups: Microsoft Controls, OpenEdge Controls, Telerik UI for WinForms, and (if installed) OpenEdge Ultra Controls.

A control group can contain any combination of controls, including controls that are also included in other control groups.

**Controls**

Controls are the objects with which you build the UI. As installed, the Toolbox includes all of the controls that Progress Software Corporation supports and that your OpenEdge license allows you to use.

A given control can be included in multiple control groups.

**Design Canvas**

The Design Canvas is the area of the design window that displays the visual container (form) and the controls you add from the Toolbox. The white pane at the top contains the WYSIWYG representation of the UI design, including all visual controls that you define. The pale yellow pane at the bottom contains non-visual controls, if applicable; this pane does not appear if no non-visual controls are in use. You can resize these panes by dragging the horizontal separator between them.

On the Design Canvas, you can place new controls, position and resize existing controls, and select controls whose properties you want to edit.

**See also**

- Using the Toolbox on page 485
- Choosing controls for the UI on page 491
- Defining properties and appearance on page 493
- Customizing the Visual Designer Toolbox on page 511

**Visual Designer Properties view**

The Properties view displays the full set of properties and events for the currently selected objects on the Design Canvas, and allows you to view and edit their values.

This view is included by default in the OpenEdge Visual Designer perspective and most other standard perspectives. To display it if it is not currently open, select Window > Show View > Properties.
The **Properties** view has two tabs, **Properties** and **Events**. Each tab includes a two-column grid that shows the name of the property or event on the left and its value, if defined, on the right.

- **Properties tab** - The technique for editing property values varies according to the specific property and the data type. Click in the right column and edit the value as appropriate. Many properties offer specialized design tools or drop-down selection lists. Many also support direct entry by typing.

- **Events tab** - An event value is the name of the corresponding event-handler method. (Double-clicking an event's name or value column automatically creates an event subscription and a handler method and switches focus to that method in an OpenEdge ABL Editor window.) You can enter or edit an event handler name by direct typing, or you can select a method from the drop-down list (which shows available methods whose signature matches that of the event).

**Note:** In addition to inherited .NET events, the **Events** tab displays the ABL events that are defined in an ABL-derived .NET class using the DELEGATE or the SIGNATURE clause. The events using the SIGNATURE clause appear in the **Events** tab only if their signature follows the .NET convention for event handler signatures. For more information on the conventions for .NET event handler signatures, see Class Events Reference in the **ABL Language Reference**.

In addition to the name-value grid, the **Properties** view includes the following elements:

<table>
<thead>
<tr>
<th>Name (unlabeled text box at top of view)</th>
<th>Displays the name of the control or form currently selected on the <strong>Design Canvas</strong>, and the class of which it is an instance. The down-arrow button to the right of the field displays a drop-down list of all objects on the <strong>Design Canvas</strong>, from which you can select a different object. Typing a letter while focus is on the name field selects the first object whose name begins with that letter.</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Sort by Categories]</td>
<td>Displays the grid with entries grouped in categories.</td>
</tr>
<tr>
<td>![Sort Alphabetically]</td>
<td>Displays the grid with entries in alphabetical order.</td>
</tr>
<tr>
<td>![Reset]</td>
<td>Resets the selected property or event to its default value.</td>
</tr>
<tr>
<td><strong>Commands panel (below grid)</strong></td>
<td>Provides links to any available design tools or custom property pages for the selected object. You can toggle the display of this panel on or off from the context (right-click) menu.</td>
</tr>
<tr>
<td><strong>Description panel (below grid)</strong></td>
<td>Provides help text for the selected property or event. You can toggle the display of this panel on or off from the context (right-click) menu.</td>
</tr>
</tbody>
</table>

**See also**

[Using the Properties view](#) on page 500  
[Coding event logic](#) on page 510

**Add Controls dialog**

The **Add Controls** dialog lets you select controls from assemblies that are installed on your system and add them to the **Toolbox**. This dialog appears when you right-click in the **Toolbox** and selects **Add Controls** from the context menu.

The **Add Controls** dialog has three tabs:
• **Browsed Assemblies** - Displays the list of controls in the assembly file that you most recently browsed during the current Progress Developer Studio for OpenEdge session.

• **Global Assemblies** - Displays the list of controls in your system's Global Assembly Cache.

• **ABL Controls** - Displays the list of user-defined controls (User Controls and Inherited Controls) in your current PROPATH.

Select one or more controls to add by clicking the check box to the left of the control name. You can sort the list by clicking a column heading.

In addition to the control list, the **Add Controls** dialog includes the following controls:

<table>
<thead>
<tr>
<th>Control Group</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control Group</td>
<td>Lets you choose the Toolbox control group to which the selected controls are to be added.</td>
</tr>
<tr>
<td>Filter</td>
<td>Lets you enter an initial character sequence by which to filter the control list. Only those controls whose name begins with the specified characters appear in the list.</td>
</tr>
<tr>
<td>Clear</td>
<td>Clears the Filter field and restores the control list to its unfiltered state.</td>
</tr>
<tr>
<td>Browse</td>
<td>Displays a standard file browser that lets you locate and select a DLL or EXE file containing valid .NET controls.</td>
</tr>
<tr>
<td>OK</td>
<td>Closes the Add Controls dialog and adds all selected controls to the Toolbox.</td>
</tr>
<tr>
<td>Cancel</td>
<td>Closes the Add Controls dialog without adding any controls to the Toolbox.</td>
</tr>
</tbody>
</table>

**See also**

Managing controls on page 513
Introducing the Class Browser

The Class Browser is a Progress Developer Studio for OpenEdge view that allows you to see the structure and content of the various classes you work with in creating ABL applications. The Class Browser's contents are organized by resources, which are collections of classes (types).

Whether you are developing with ABL or .NET classes, you can use the Class Browser's three-panel display to easily see details about all your ABL and .NET resources, packages or namespaces, and types. By default, the Class Browser displays this information for all resources, whether within or outside of a project in the workspace. For r-code based on class files and Progress procedure libraries within a project, the details the Class Browser displays are derived from the PROPATH associated with each open project in the workspace. (Directories and procedure libraries that do not have r-code classes are not displayed by the Class Browser.) For .NET classes, the details come from the assemblies associated with each project in the workspace. The same level of information can be displayed for external resources you add to the Class Browser.

Depending on which item is in focus in the Class Browser, the details you view can also include information about properties, data members, constructors, methods, and events. The Class Browser displays type information for .NET classes using summary text and ABL syntax.
For ABL classes, you can see type information, as well as similarly helpful details about membership, inheritance, and sample ABL syntax for working with the USING and DEFINE statements. You can also see which ABL classes, events, properties, and methods are abstract.

The Class Browser is an Eclipse view. For quick access, the OpenEdge Editor, OpenEdge AppBuilder, and OpenEdge Visual Designer perspectives include the Class Browser as a Fast View.

Viewing ABL and .NET types

For ABL types, the set of classes to browse are available for each PROPATH and Progress procedure library file for all the projects in the workspace. Conversely, the classes for all .NET types are listed in the assemblies.xml file for each project in the workspace. In the Class Browser, you see the ABL types organized into packages. .NET types are organized into namespaces.

The following assemblies are included by default since they are commonly used by OpenEdge GUI for .NET applications:

- mscorlib.dll
- System.dll
- System.Drawing
- System.Windows.Forms

These assemblies are available to each project and can be added to the assemblies.xml file. Even when all projects for the workspace are closed, these assemblies appear in the Class Browser view.

Information in the Class Browser view is read-only; however, you can copy and paste code templates to use as program code outside of the view.

Browse modes

You can display information in the Class Browser in either of the following modes:

- Browse by resource - To see an alphabetical list of all resources.
- Browse by project - To see a list of open projects. Within each project node, the resources are displayed alphabetically.

The browse mode you are using when you exit the Class Browser persists for the workspace to the next Progress Developer Studio for OpenEdge session.

External resources

In the Class Browser, you view details about all classes (for the projects in a workspace) that are in the included PROPATHs (procedure libraries and directory paths) and assemblies. The selection of classes whose details you can display is dependent on the Class Cache preference settings you chose in Progress Developer Studio for OpenEdge for the entire workspace. For details, see the description of Class Cache options in the ABL Editor online help.

You can also add and then view details about external resources, such as assemblies, Progress procedure library files, and path entries. The assemblies you add can be either local assemblies or assemblies registered in the Global Assembly Cache (GAC).

For details, see the following topics:

- Concepts
- Tasks
- Reference
Concepts

The Class Browser view

The Class Browser view comprises three main sections—the **Browser** pane, the **Summary** pane, and the **Content** pane. Like many other views, the Class Browser also includes a toolbar.

The panes display information in the following way:

- **The Browser pane** displays the available resources: assemblies, Progress procedure libraries (provided they contain r-code classes), and paths.

- **The Summary pane** displays a description of the selected resource, package or namespace, type, or member.

- **The Content pane** displays a list of the members contained in a type (in a class or interface, for example).

From the toolbar, you can control whether the Class Browser displays information by resource or project. You can also navigate back or forward; add an external resource (such as a path, Progress procedure library, or assembly); collapse all tree nodes; and filter the member information you want to view.

See also

- [Introducing the Class Browser](#) on page 533
- [Accessing the Class Browser](#) on page 536
- [Viewing information in the Class Browser](#) on page 537
- [Adding external resources](#) on page 539
- [Browser Pane](#) on page 542
- [Content Pane](#) on page 542
- [Summary Pane](#) on page 543
External resources and the Class Browser

You can add external resources, such as assemblies, Progress procedure library files, and path entries, to the Class Browser and then view information about their types. You can also add local assemblies or assemblies registered in the Global Assembly Cache (GAC) as external resources. When you add an external resource, the Class Browser writes to the `classBrowser.xml` file for the workspace, updating the file to include details about the new resource.

You add external resources only once to the Class Browser. External resources that you add to the Class Browser persist between sessions of Progress Developer Studio for OpenEdge.

If you close and then reopen the Class Browser during the same Progress Developer Studio for OpenEdge session, the most recently used browse mode is retained.

With the Class Browser in the browse by resource mode, you can remove external resources that you added.

See also

Viewing information in the Class Browser on page 537
Adding external resources on page 539
Add External Resources dialog on page 548

Tasks

Accessing the Class Browser

You can access the Class Browser by any of the following methods:

- Click Class Browser from the Fast View toolbar.
- From the OpenEdge Editor, OpenEdge AppBuilder, OpenEdge Server, or OpenEdge Visual Designer perspective, click Fast View. Choose Class Browser from the list.
- From any Progress Developer Studio for OpenEdge perspective, press CTRL+F12. Depending on the perspective, the Class Browser opens either as a Fast View (from the OpenEdge Editor, OpenEdge AppBuilder, OpenEdge Server, or OpenEdge Visual Designer) or as you have configured it as a view.
- From the menu bar, choose Window > Show View > Class Browser.

The Class Browser view opens and displays the most recently used browse mode, either by resource (the default) or project. The Browser pane displays all nodes as they existed when the view was last opened, with the most recently selected type in focus. The Content pane and the Summary pane are updated with the corresponding information.

If you open the Class Browser as a Fast View, you can close it by pressing ESC.

See also

The Class Browser view on page 535
Viewing information in the Class Browser on page 537
Browser pane on page 542
Viewing information in the Class Browser

You can use either of two modes to view information in the Class Browser:

- **Browse by Resource**: Displays all resources in an overall view. You are not required to have an OpenEdge project available to browse by resource.

- **Browse by Project**: Displays all resources by project. ABL built-in types and mscorlib.dll are also shown for each project node when you browse by project.

  The **Browse by Project** mode does not display external resources.

To browse by resource, click **Browse by Resource** in the Class Browser toolbar; to browse by project, click **Browse by Project**.

**See also**
- The Class Browser view on page 535
- Browsing by resource on page 537
- Browsing by project on page 538
- Browser pane on page 542
- Content pane on page 542
- Summary pane on page 543
- Class Browser view icons on page 546

**Browsing by resource**

You can browse by resource to see a list of assemblies, procedure libraries, or paths as top-level nodes. When you expand the nodes, the ABL packages or .NET namespaces appear as second-level nodes. Expand these second-level nodes to see the third level of contents -- the types (the classes and/or interfaces).

You are not required to have an OpenEdge project available to browse by resource, which can be helpful if you are browsing external resources.

1. To begin, click **Browse by Resource** on the Class Browser toolbar. A list of top-level resources, such as assemblies, procedure libraries, and paths, appears in the Browser pane, and the Content and Summary panes are empty.

2. You can view the second and third levels of a top-level resource in either of the following ways:

   - Click the name of the top-level resource in the **Browser pane**. Its second-level package or namespace nodes appear in a list in the Content pane; summary details about the resource appear in the **Summary pane**. Click a second-level package or namespace node in the **Content pane** to see details about it in the Summary pane.
Chapter 12: Introducing the Class Browser

- Expand the node of a top-level resource in the Browser pane. Its second-level package or namespace nodes appear below it in the Browser pane. Click the name of the package or namespace to view a list of its classes and interfaces in the Content pane; or expand the package or namespace to view the class and interface list in the Browser pane.

3. Click the name of a class or interface in the Browser pane to see its members in the Content pane and its summary in the Summary pane.

See also
The Class Browser view on page 535
Browsing by project on page 538
Browser pane on page 542
Content pane on page 542
Summary pane on page 543
Class Browser view icons on page 546

Browsing by project

Browse by project to see a list of open Progress Developer Studio for OpenEdge projects as top-level nodes. (No nodes appear if there are no available projects.) When you expand the nodes, the various resources specified by the project are displayed as second-level nodes. Expanding the second-level resources brings third-level packages and namespaces into the view.

Note: The Browse by Project mode does not display external resources.

1. To begin, click Browse by Project in the Class Browser toolbar. A list of available Progress Developer Studio for OpenEdge projects appears in the Browser pane, and the Content and Summary panes are empty.

2. You can view the contents of a project in either of the following ways:

- Click the name of the project in the Browser pane. Its contents appear in a list in the Content pane; summary details about the project appear in the Summary pane. Click one of the items in the Content pane to see details about it in the Summary pane. For the mscorlib.dll, for example, the Summary pane identifies it as an assembly and provides its location.

- Expand the project node in the Browser pane. Its contents appear listed as nodes below it in the Browser pane. Click the node to view a list of its packages and namespaces in the Content pane; or expand the node to see packages or namespaces in the Browser pane. Click the package or namespace (in the Browser pane or the Content pane) to see a list of its classes and interfaces. Click a class or interface to see its members. The Summary pane provides details for the item in focus.

See also
The Class Browser view on page 535
Browsing by resource on page 537
Browser pane on page 542
Content pane on page 542
Summary pane on page 543
Class Browser view icons on page 546
Showing members

Each class consists of members. The members differ as follows:

- **Public members** - Can be accessed by the class defining them, by any class that inherits from that class, and by other classes and procedures that reference an instance of that class.

- **Protected members** - Can be accessed by the class defining them and by any class that inherits from that class.

In addition, members can also be inherited, meaning they are available from the superclass.

1. To choose which members you want the Class Browser to display, do either of the following:
   - From the Content pane, right-click to see the context menu.
   - From the Class Browser toolbar, click Menu.

2. Choose one or more of the available options: **Show Public Members**, **Show Protected Members**, **Show Inherited Members**. (All three options are initially selected by default.)

The Content pane displays only those members you have chosen to view. You cannot display private members in the Class Browser, since they cannot be used outside of the defining class.

See also

- [The Class Browser view](#) on page 535
- [Browsing by resource](#) on page 537
- [Browsing by project](#) on page 538
- [Browser pane](#) on page 542
- [Content pane](#) on page 542
- [Summary pane](#) on page 543
- [Class Browser view icons](#) on page 546

Adding external resources

You can add external resources, such as assemblies, Progress procedure library files, and path entries, to the Class Browser and view their type information. The assemblies you add can be either local assemblies (DLLs or EXEs) or assemblies registered in the Global Assembly Cache (GAC).

You can add an external resource only once to the Class Browser. External resources that you add to the Class Browser persist between sessions of Progress Developer Studio for OpenEdge.

When you add an external assembly reference, the Class Browser automatically updates the classBrowser.xml file for the workspace.

Follow the given steps:

1. To add one or more external resources, do one of the following:
   - From the Browser pane, right-click and choose **Add External Resources** from the context menu.
   - From the Class Browser toolbar, click **Add External Resources**.

      The **Add External Resources** dialog appears.

2. To add a path or a procedure library (.pl):
a) Choose the R-code path/PLs tab.

b) Provide a path or locate the procedure library you want to add. (You can add multiple resources.)

c) Click OK. The Add External Resources dialog closes, and the resource is added.

3. To add an assembly from the GAC:

a) Choose the Global Assemblies tab. Select one or more (use CTRL+CLICK) assemblies to add.

b) Click OK. The Add External Resources dialog closes. The resource or resources are added and appear in the Browser pane.

4. To add a local assembly:

a) Choose the Local Assemblies tab. Provide a path or locate the assembly file or executable file you want to add.

b) Click OK.

See also

External resources and the Class Browser on page 536
Removing external resources on page 540
Add External Resources dialog on page 548

Removing an external resource

You can remove an external resource from the Class Browser. To do this, you must be browsing by resource (rather than by project).

If an external resource was also defined by a project, the resource will still be listed in the Class Browser; however, the Remove External Resources option will be disabled.

To remove external resources:

1. In the Browser pane, select one or more external resources you want to remove.

2. Right-click, and choose Remove External Resources from the context menu. A message appears asking you to confirm the removal of the resource(s).

3. Click OK. The Class Browser view refreshes, and the external resources are no longer listed.

See also

External resources and the Class Browser on page 536
Add External Resources dialog on page 548
Browser pane on page 542

Searching in the Class Browser

You can search the Class Browser for types, packages, or namespaces by specifying a partial name. (You cannot search for member information, including events, methods, or property names.)

To search in the Class Browser:
1. From the Browser pane, type the text you want to search for in the Search field.

2. Click **Search**. If the search produces results, the matching types with fully qualified names appear in the Browser pane. If the search produces no results, a message appears telling you that no matches were found.

Once a search completes, you can restore the previous content of the Browser pane by clearing the Search field or clicking **Clear Search**.

The Class Browser does not retain search results between sessions.

**See also**

Browser pane on page 542

### Copying to the Clipboard

You can copy text from the Class Browser view and paste it to the Clipboard for use outside of the Class Browser, as follows:

- From the Browser pane, you can copy the qualified name of the selected resource, project, package, namespace, or type.
- From the Content pane, you can copy the name of the selected group or member, a method signature, or the signature of an indexed property.
- From the Summary pane, you can copy the selected text.

You can copy one or more items at a time.

To copy:

1. Select the text and right-click. To copy more than one item, use **CTRL+CLICK** to select each item.
2. From the context menu, choose **Copy**. (From the Summary pane, you can also use **CTRL+C** to copy the selection.) If you selected multiple items, a comma-delimited list of the item names is copied to the Clipboard.

**Note:** You must have at least one item selected for the **Copy** option to be available.

**See also**

The Class Browser view on page 535

Viewing information in the Class Browser on page 537

Browser pane on page 542

Content pane on page 542

Summary pane on page 543
Reference

Browser pane

The Browser pane displays the selected assemblies, procedure libraries, and paths (resources). The information is organized into three levels:

- Resources
- Packages (ABL) and namespaces (.NET)
- Types (classes or interfaces)

When you expand a resource node, its packages or namespaces are listed. If you expand the package or namespace node, the types contained within are listed. Generic type parameters are listed for the corresponding generic type.

When you select one of these types, the Content pane displays its constructors, methods, properties, events, and data members. Generic arguments also appear, as applicable, in the Content and Summary panes. If you select multiple types in the Browser pane, the Content pane and the Summary pane are empty.

If a type does not have a package or namespace, it is listed under the default package node when package names are displayed in the Browser pane.

See also
- Class Browser view on page 535
- Browsing by resource on page 537
- Browsing by project on page 538
- Searching in the Class Browser on page 540
- Content pane on page 542
- Summary pane on page 543

Content pane

The Content pane displays a list of the contents of a resource's packages or namespaces, a list of the package or namespace's types, or a list of the type's members. Members are grouped by type and listed in alphabetical order. When you select a type in the Browser pane, all member information for the type appears in the Content pane. If you select multiple resources in the Browser pane, the Content pane is empty.

The pane also displays extent or generic information for method parameters. (Extent and generic information does not appear for variables and properties.)

Note: The Content pane does not display private members.

For more information about a member method, choose a specific method in the Content pane. Details about the method appear in the Summary pane.

The Content pane's context menu includes three display options: Show Public Members, Show Protected Members, and Show Inherited Members. All three options are selected by default.
The Content pane displays... | When you select this item in the Browser pane...
---|---
The resources associated with the selected project | A project
The packages or namespaces for the types contained in the selected resource | A resource
The types for the selected package or namespace within the resource | A package or a namespace
The members of the selected type: data members, constructors, methods, properties, and events | A type

Using the **Copy** option, you can copy the signature of a method or the reference to a property or variable and paste it elsewhere, such as in the ABL Editor.

**See also**

- Class Browser view on page 535
- Viewing information in the Class Browser on page 537
- Browser pane on page 542
- Summary pane on page 543

**Summary pane**

The Summary pane displays a description for the selected resource, package, namespace, type, or member. For members of a type, the pane includes sample ABL syntax that you can use to access the selected member. If there is a mapped data type, it is shown in parentheses.

The pane also displays extent information for methods, properties, and variables that are arrays. In addition, abstract classes, methods, properties, and events are identified as such in the description of the element.

You can select any text in the Summary pane and copy it to the clipboard for use outside of the Class Browser. If you select multiple types in the Browser pane or the Content pane, the Summary pane is empty.

The information that the Summary pane displays for .NET types is extracted from the .xml file associated with the assembly file. For example, the documentation for the types in `mscorlib.dll` is included in `mscorlib.xml`.

<table>
<thead>
<tr>
<th>When you select this item in the Browser pane . . .</th>
<th>The Summary pane displays . . .</th>
</tr>
</thead>
</table>
| An assembly, procedure library, or path entry | • The name of the resource.
• The location of (path to) the resource.
• For an external resource, internal resource, or if selected by project: a list of project names that reference the resource.
• A summary (description) of the resource, if available. |
<table>
<thead>
<tr>
<th>When you select this item in the Browser pane . . .</th>
<th>The Summary pane displays . . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>A project</td>
<td>• The name of the project.</td>
</tr>
<tr>
<td></td>
<td>• Path to the project directory.</td>
</tr>
<tr>
<td>A package or a namespace</td>
<td>• The name of the package or namespace.</td>
</tr>
<tr>
<td></td>
<td>• The list of resources for which the package or namespace is a member.</td>
</tr>
<tr>
<td></td>
<td>• A summary (description) of the package or namespace.</td>
</tr>
<tr>
<td>A type</td>
<td>• The name of the type.</td>
</tr>
<tr>
<td></td>
<td>• Whether it is FINAL or ABSTRACT.</td>
</tr>
<tr>
<td></td>
<td>• The list of packages or namespaces for which the type is a member.</td>
</tr>
<tr>
<td></td>
<td>• The name of the base type it inherits from.</td>
</tr>
<tr>
<td></td>
<td>• The list of interfaces being implemented.</td>
</tr>
<tr>
<td></td>
<td>• A summary (description) of the type, if available.</td>
</tr>
<tr>
<td></td>
<td>• Examples of ABL syntax using the type.</td>
</tr>
<tr>
<td>A data member</td>
<td>• The type of data member.</td>
</tr>
<tr>
<td></td>
<td>• The name of the data member.</td>
</tr>
<tr>
<td></td>
<td>• ABL syntax showing the data member definition, which includes access modifier and type.</td>
</tr>
<tr>
<td></td>
<td>• The name of the type that contains the data member.</td>
</tr>
<tr>
<td></td>
<td>• A summary (description) of the data member, if available.</td>
</tr>
<tr>
<td>A method</td>
<td>• The name of the method and details about it, including PUBLIC, PROTECTED, FINAL, OVERRIDE, STATIC, or ABSTRACT.</td>
</tr>
<tr>
<td></td>
<td>• ABL syntax showing the method definition, which includes the access modifier, the method name, the method's parameters, and the return value.</td>
</tr>
<tr>
<td></td>
<td>• The name of the type that contains the method.</td>
</tr>
<tr>
<td></td>
<td>• A summary (description) of the method, if available.</td>
</tr>
<tr>
<td></td>
<td>Also shown if applicable:</td>
</tr>
<tr>
<td></td>
<td>• A description of the return values of the method (if available).</td>
</tr>
<tr>
<td></td>
<td>• Parameters (if available).</td>
</tr>
<tr>
<td>When you select this item in the Browser pane . . .</td>
<td>The Summary pane displays . . .</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>---------------------------------</td>
</tr>
</tbody>
</table>
| An event                                         | • The name of the event, including an indication of whether the event is abstract.  
• ABL syntax showing the event definition, which includes the access modifier, event name, and the event's parameters, as well as a method definition for a method handler.  
• The name of the type that contains the event.  
• A summary (description) of the event, if available.  
• Examples of ABL syntax using the event.  
• The signature of the event. The SIGNATURE keyword appears for ABL events that include a signature, and the DELEGATE keyword appears for ABL events that use a delegate.  
Summary information for .NET events is the same in structure as for ABL events and also shows the signature of the event.  
Also shown if applicable:  
• A description of the parameters (if available) |
| A property                                       | • The name of the property, including an indication of whether the property is abstract.  
• The data type.  
• The parameters, if applicable, shown in brackets for an indexed property.  
• Accessor-access-mode (if valid).  
• The name of the type that contains the property.  
• A summary (description) of the property, if available.  
• A note that indicates if a property is an indexed property.  
• Examples of ABL syntax using the property.  
Also shown if applicable:  
• A description of the parameters (if available). |

**See also**

- [Class Browser view](#) on page 535
- [Viewing information in the Class Browser](#) on page 537
- [Browser pane](#) on page 542
- [Content pane](#) on page 542
Class Browser toolbar

The Class Browser toolbar allows you to do the following within the Class Browser view:

- Navigate back or forward through recently selected items.
- View information by resource or project.
- Add external resources.
- Collapse all expanded nodes in the Browser pane.
- Refresh the view.
- Choose which members to view.
- Minimize or maximize the view.

The toolbar contains the following buttons:

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="" alt="Icon" /></td>
<td>Navigate back through recently selected items.</td>
</tr>
<tr>
<td><img src="" alt="Icon" /></td>
<td>Navigate forward through recently selected items.</td>
</tr>
<tr>
<td><img src="" alt="Icon" /></td>
<td>Browse by resource.</td>
</tr>
<tr>
<td><img src="" alt="Icon" /></td>
<td>Browse by project.</td>
</tr>
<tr>
<td><img src="" alt="Icon" /></td>
<td>Add one or more external resources.</td>
</tr>
<tr>
<td><img src="" alt="Icon" /></td>
<td>Collapse all expanded tree nodes in the Browser pane.</td>
</tr>
<tr>
<td><img src="" alt="Icon" /></td>
<td>Refresh the Class Browser view.</td>
</tr>
<tr>
<td><img src="" alt="Icon" /></td>
<td>Choose whether to show public members, show protected members, and/or show inherited members.</td>
</tr>
<tr>
<td><img src="" alt="Icon" /></td>
<td>Minimize the Class Browser view.</td>
</tr>
<tr>
<td><img src="" alt="Icon" /></td>
<td>Maximize the Class Browser view.</td>
</tr>
</tbody>
</table>

See also

Class Browser view on page 535
Viewing information in the Class Browser on page 537
Browser pane on page 542
Content pane on page 542
Summary pane on page 543

Class Browser view icons

The following icons help you identify items in the Class Browser view:
<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Class Browser view." /></td>
<td>Class Browser view.</td>
</tr>
<tr>
<td><img src="image" alt="Browse by resource." /></td>
<td>Browse by resource.</td>
</tr>
<tr>
<td><img src="image" alt="Browse by project." /></td>
<td>Browse by project.</td>
</tr>
<tr>
<td><img src="image" alt="Add external resources." /></td>
<td>Add external resources.</td>
</tr>
<tr>
<td><img src="image" alt="Assembly." /></td>
<td>Assembly.</td>
</tr>
<tr>
<td><img src="image" alt="Progress Procedure Library." /></td>
<td>Progress Procedure Library.</td>
</tr>
<tr>
<td><img src="image" alt="Path entry." /></td>
<td>Path entry.</td>
</tr>
<tr>
<td><img src="image" alt="ABL built-in types." /></td>
<td>ABL built-in types.</td>
</tr>
<tr>
<td><img src="image" alt="Search." /></td>
<td>Search.</td>
</tr>
<tr>
<td><img src="image" alt="Clear search." /></td>
<td>Clear search.</td>
</tr>
<tr>
<td><img src="image" alt="Class." /></td>
<td>Class.</td>
</tr>
<tr>
<td><img src="image" alt="Interface." /></td>
<td>Interface.</td>
</tr>
<tr>
<td><img src="image" alt="Constructor tree." /></td>
<td>Constructor tree.</td>
</tr>
<tr>
<td><img src="image" alt="Constructor." /></td>
<td>Constructor. <em>In this case, the green dot indicates that the constructor is public. See the following table for additional clarification.</em></td>
</tr>
<tr>
<td><img src="image" alt="Method tree." /></td>
<td>Method tree.</td>
</tr>
<tr>
<td><img src="image" alt="Method." /></td>
<td>Method.</td>
</tr>
<tr>
<td><img src="image" alt="Event tree." /></td>
<td>Event tree.</td>
</tr>
<tr>
<td><img src="image" alt="Event." /></td>
<td>Event.</td>
</tr>
<tr>
<td><img src="image" alt="Data member tree." /></td>
<td>Data member tree.</td>
</tr>
<tr>
<td><img src="image" alt="Data member." /></td>
<td>Data member.</td>
</tr>
<tr>
<td><img src="image" alt="Property tree." /></td>
<td>Property tree.</td>
</tr>
<tr>
<td><img src="image" alt="Property." /></td>
<td>Property.</td>
</tr>
</tbody>
</table>

To help you further distinguish what you see in the Class Browser view, an icon representing an item may be accompanied by one or more additional indicators:
If you see this additional indicator... | The item is... | Example
---|---|---
A green dot | Public | ![Capacity](Capacity.png)
A yellow diamond | Protected | ![TextProtectedMethod](TextProtectedMethod.png)
The letter S | Static | ![Adapter(System.Collections_IList)](Adapter(System.Collections_IList).png)
The letter C | A constructor | ![TextProperties](TextProperties.png)

**Note:** In this example, the green dot indicates that the item is also public.

The letter F | Final | ![DictionaryBase](DictionaryBase.png)
A red box containing a white letter X | Not available | ![SourceType[NotAvailable]](SourceType[NotAvailable].png)

### See also
- [Class Browser view](#) on page 535
- [Viewing information in the Class Browser](#) on page 537
- [Browser pane](#) on page 542
- [Content pane](#) on page 542
- [Summary pane](#) on page 543

### Add External Resources dialog

The **Add External Resources** dialog appears when you:

- Right-click in the Browser pane and choose **Add External Resources** from the context menu.
- Click **Add External Resources** ![Add External Resources](AddExternalResources.png) from the Class Browser toolbar.

From this dialog you can add to the Class Browser view one or more resources from the following categories:

<table>
<thead>
<tr>
<th>R-code path/PLs</th>
<th>A path or a Progress procedure library</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Global Assemblies</strong></td>
<td>An assembly from the Global Assembly Cache</td>
</tr>
<tr>
<td><strong>Local Assemblies</strong></td>
<td>An assembly or an executable file</td>
</tr>
</tbody>
</table>

### See also
- [Adding external resources and Class Browser](#) on page 536
- [Adding external resources](#) on page 539
- [Removing external resources](#) on page 540
Browser pane on page 542
Running and debugging ABL programs

Progress Developer Studio for OpenEdge provides support for running and debugging programs using launch configurations. A launch configuration determines the characteristics of the ABL Virtual Machine (AVM) session in which the program runs.

For a debugging session, in addition to starting the appropriate launch configuration, Progress Developer Studio for OpenEdge starts its Debugger.

The topics in this help module explain:

- How to manage and use launch configurations
- How to use the Progress Developer Studio for OpenEdge Debugger

For details, see the following topics:

- Concepts
- Tasks
- Reference

Concepts

Launch configurations for running and debugging programs

You can use a launch configuration in the Progress Developer Studio for OpenEdge to:
Chapter 13: Running and debugging ABL programs

- Run and debug ABL programs.
- Run and debug AppServer or WebSpeed instances (only for version 11.0 or later).
- Attach the Debugger to an external AVM (only for version 11.0 or later).

A launch configuration defines the characteristics of the AVM instance under which the selected program runs. These characteristics include such things as startup parameters, PROPATH settings, and environment settings for the AVM session; database connections; and whether the program uses a dedicated instance of the AVM or the instance under which your OpenEdge project is currently running.

You can use the Configurations wizard to define all of a launch configuration's characteristics. Although this wizard contains a large number of fields on multiple tabs, defining a launch configuration need not be a complicated task. In fact, with a single click, you can create and run a launch configuration that uses default settings, and then edit any of these settings, if necessary.

Note: The descriptions of launch configuration behavior in this online help apply to an Eclipse installation created by the OpenEdge installation program, with default preference settings. You can adjust certain aspects of launching behavior by navigating to Window > Preferences > Run/Debug/Launching.

This section provides some basic information about how launch configurations work.

See also
Running and debugging ABL programs on page 551

Accessing launch configuration functions

To open the Configurations dialog, create a default launch configuration, or run an existing configuration, you can either use the Run and Debug buttons on the main toolbar, or use the ABL Editor context (right-click) menu.

Toolbar buttons

The Run and Debug buttons look like this:

<table>
<thead>
<tr>
<th></th>
<th>Run</th>
<th>Debug</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="image" alt="Run" /></td>
<td><img src="image" alt="Debug" /></td>
</tr>
</tbody>
</table>

- Clicking the button image itself runs the currently selected program or starts it in the Debugger, creating a default launch configuration if necessary.

- Clicking the drop-down arrow opens a menu from which you can start an existing configuration, create and run a default configuration, or open the Configurations wizard for access to all options. For quick access to existing configurations, the top level of this menu includes recently used configurations. You can also choose to add any of your configurations to the Favorites section of this menu.

Servers view

The same two buttons appear on the toolbar of the Servers view, where you can use them interchangeably with those on the main toolbar to start the currently selected AppServer or WebSpeed instance. The Servers view buttons do not include the down arrow, so click the arrow on the main toolbar to open the menu.

Launch configurations > Run As > Context menu options
• The ABL Editor context menu provides Run As and Debug As options, each of which has a sub-menu from which you can choose one of the following options:

• Run as or Debug as - Runs or debugs the currently open program, creating a default launch configuration if necessary

• Run Configurations or Debug Configurations - Opens the Configurations wizard for access to all options

See also
Progress Developer Studio for OpenEdge Debugger on page 554
Using Launch Configurations on page 562
Launch Configuration Settings on page 589

Essential launch configuration settings

Of the many options that it is possible to specify for a launch configuration, two have particular importance:

• Project - You must associate the configuration with an existing project in your workspace. This setting does not prevent you from using the configuration in other projects.

• Startup program - You define the configuration either to run a specific program, or to run the program that currently has focus in your workspace (selected in the Project Explorer view or open in the ABL Editor, whichever applies).

OpenEdge uses these two settings to determine the behavior of the Run (As) and Debug (As) commands, as explained in the next topic.

Note: These options do not apply to AppServer or WebSpeed launch configurations.

See also
How OpenEdge creates or reuses a configuration on page 553
Accessing launch configuration functions on page 552

How OpenEdge creates or reuses a configuration

Although you can explicitly select an existing launch configuration to run or debug a program, in many cases, the easiest approach is simply to click Run or Debug with focus on the desired startup item. (In most cases, clicking the button produces the same result as opening the menu and selecting Run As or Debug As.)

When you choose one of these options, OpenEdge determines whether a suitable configuration already exists, and if so, runs that configuration. If a suitable configuration does not exist, OpenEdge creates and runs a new configuration. The basis for identifying a suitable configuration depends on whether the selected item is an ABL source file or an AppServer or WebSpeed instance.

For an ABL program
To determine whether to use an existing configuration to run or debug an ABL program, OpenEdge performs the following tests in sequence:

• If a configuration exists whose Startup program setting specifically identifies the selected program and whose Project setting matches the project of the selected program, OpenEdge uses that configuration. Otherwise, test 2 is performed.
• If a configuration exists whose Startup program setting does not identify a specific program (but instead refers generically to the "selected program") and whose Project setting matches the project of the selected program, OpenEdge uses that configuration.

If multiple existing configurations satisfy a test, you are prompted to choose one of those configurations.

If both tests fail, OpenEdge creates and runs a new configuration with default settings. You can edit this configuration as appropriate.

For AppServer and WebSpeed instances

To determine whether to use an existing configuration to run or debug an AppServer or WebSpeed instance, OpenEdge looks for a configuration specifying that server. If such a configuration exists, OpenEdge uses it to start the AppServer or WebSpeed server. (OpenEdge automatically creates a default launch configuration when each AppServer or WebSpeed instance is defined, so a server has at least one launch configuration unless you delete it.)

If multiple configurations specify the same server, the one that was defined first is used.

If no configuration exists for the specified server, Progress Developer Studio for OpenEdge creates and runs a new configuration with a name matching the server name.

See also

Using Launch Configurations on page 562
Using the OpenEdge Architect Debugger on page 569
Launch Configuration Settings on page 589

Progress Developer Studio for OpenEdge Debugger

The Debugger is an integral component of Progress Developer Studio for OpenEdge. It is an interactive tool that provides a full set of tools for testing and debugging programs. It works directly with the ABL Editor, making it easy to identify and fix problems as you work.

The following are the key features of the debug launch configurations provided in the Progress Developer Studio for OpenEdge:

• AppServer and WebSpeed debugging - Progress Developer Studio for OpenEdge supports debugging of remote and local AppServer and WebSpeed instances.

• Attachable Debugger - The debug launch configuration type allows you to attach the Debugger to an external AVM (which can be an ABL client, a WebSpeed agent, or an AppServer running on the same machine or a remote machine).

• AppServer pass-through - You can debug a remote procedure from within Progress Developer Studio for OpenEdge, during pass-through, without having to set breakpoints in the remote procedure code.

• SpeedScript debugging - Progress Developer Studio for OpenEdge allows you to debug .w files that are generated from HTML files in a SpeedScript editor.

• Standalone Debugger functionality - Progress Developer Studio for OpenEdge provides all the debugging features that are available with the standalone OpenEdge Debugger.

You can use the Debugger in Progress Developer Studio for OpenEdge to run multiple debugging sessions. Each session uses a dedicated instance of the OpenEdge runtime engine, and provides these benefits:

• Debugging multiple programs simultaneously, whether the programs are in the same project or in different projects.

• Debugging multiple instances of the same program.
• Editing the source code using all Editor features, while a debugging session is active.
• Stepping through the code, line by line, or stop at specified lines and then resume execution from those points.
• Examining data from the procedure you are running, including the names and contents of variables it defines.
• Attaching the Debugger to a running process.
• Debugging programs running on a remote machine.
• Debugging AppServer and WebSpeed instances running on a remote machine or a local machine.

The standalone OpenEdge Debugger
In addition to the Debugger provided by Progress Developer Studio for OpenEdge, OpenEdge provides a standalone Debugger that runs with any OpenEdge application.

To launch the standalone OpenEdge Debugger, select Debugger from the Progress OpenEdge program group on the Windows Start > Programs menu. For more information about the standalone OpenEdge Debugger, see OpenEdge Development: Debugging and Troubleshooting in the Product Documentation section of the Progress Software Developer's Network Website.

Note: You must use the standalone OpenEdge Debugger for debugging an AppServer version prior to 11.0.

For the purposes of this online help, however, all references to "the OpenEdge Debugger" or "Debugger" pertain to the Debugger in Progress Developer Studio for OpenEdge.

The OpenEdge Debugger perspective
In its default state, the Debug perspective partitions the Workbench into four functional areas:

• Program execution area - Contains the Debug view, which displays session trees with stack traces for each active debugging session. This view also provides the commands for controlling program execution.
• Editing area - Contains one or more ABL Editor tabs for viewing and modifying the source code.
• Execution monitoring area - Contains the Variables, Expressions, and Dynamic Objects views, for tracking the values of data elements as the application runs. The Breakpoints view, where you define rules for suspending program execution at specific points, is also found here.
• Information area - Contains the standard Eclipse Console and Task views.

Note: An Eclipse preference setting lets you control whether every call to an external procedure or include file opens a new ABL Editor window to display the file, or whether all called files use the same window. Select Window > Preferences > Run/Debug and check or clear the Reuse editor when displaying source code option according to your preference. If you check this option, no more than two ABL Editor windows will open automatically during a debugging session -- one window for the initial file, and a second for all called files.

See also
Debugging Options on page 559
Using the Progress Developer Studio for OpenEdge Debugger on page 569
Debugger Views on page 624
Prerequisites for using the Debugger

Debugging is automatically enabled for you in Progress Developer Studio for OpenEdge. If debugging has been disabled for some reason, however, you must enable it. You can do so in any of the following ways:

• Run a command-line utility.
• Set an environment variable (at the operating system level).
• Choose to enable debugging when you begin your debugging session.

For details about each of these options, see Enabling debugging.

Additional prerequisites for using the Debugger include:

• You must make compiled r-code available for all files to be debugged. You can specify in your settings whether .r files are automatically created when you compile code, and where the files are stored.

• You must make sure that the database connections used by your application allow multiple users; that is, that the connection parameters do not include the -1 option. See .

• You must have access to the source code for the program running on an external AVM outside Progress Developer Studio for OpenEdge, in order to debug it.

Although optional, it is a good idea to turn on the display of line numbers in the ABL Editor. To do so, select Window > Preferences > General > Editors > Text Editors, and check the Show line numbers option.

See also
Enabling debugging on page 570
Configuring r code generation on page 571

OpenEdge Debugger preferences

The OpenEdge Debugger preferences are categorized into two: General and Communication. You can optionally use these preferences to manage your Debugger environment better.

General preferences

• Enable property evaluation: This preference gives you the option to choose whether you want the evaluation of properties with accessors to automatically occur. The preference is disabled by default. If you enable this preference, you see the actual value of the property in the Variables view. (A property added to the Expressions view is always evaluated, whether or not you select this preference.)

• Always select first matching file upon CRC mismatch: This preference allows you to choose whether to always select the first matching file and ignore the rest, if there are one or more files with a similar name but none of them match with the Cyclic Redundancy Check (CRC) of the program being debugged.

Communication preference

Connection timeout: This value indicates how long, in milliseconds, Progress Developer Studio for OpenEdge waits to connect to the OpenEdge Client. The value is set by default to 20000, but you can modify it to your preference, provided the value is no less than zero and no greater than 3600000 (one hour).

See also
Setting Debugger Preferences on page 569
Debug Preferences on page 619
Remote debugging support in Progress Developer Studio for OpenEdge

Remote debugging is the ability to debug server-side procedures and remote-application procedures. This mode prevents debugging any procedure in a client application session. It allows you to debug application code running on an AppServer process or a WebSpeed agent independently of the client application invoking it. Remote debugging is useful for debugging remote procedures that are:

- Running on an AppServer
- Running on a WebSpeed agent
- Executed by an Open Client (non-ABL client) application
- Executed by an OpenEdge Web service
- Executed asynchronously on the AppServer from an ABL client application

Progress Developer Studio for OpenEdge enables you to debug an external AVM (which can be an ABL application, an AppServer or a WebSpeed instance) remotely by specifying the host name and the port number. You can also step into the remote procedure code on the AVM invoked by the client application. This feature requires that you have access to the ABL source code being executed on the attached AVM, AppServer or WebSpeed broker.

This section provides some basic information about remote debugging support in Progress Developer Studio for OpenEdge.

Attachable Debugger

You can use a debug launch configuration type in the Progress Developer Studio for OpenEdge to attach the Debugger to a running client process. The client can be an ABL client, a WebSpeed agent, or an AppServer running on the same machine or a remote machine. The debug launch configuration type allows you to specify the connection type and properties of the remote or local external AVM to which you want to attach the Debugger. For more information, see Progress External OpenEdge AVM settings (Debug Configuration only) on page 615.

To attach the Debugger to an external AVM running on another machine, you can use the debug launch configuration type and specify the host name and the port number of the external AVM. To attach it to a locally running AVM, you can specify its process ID and port number or choose from a list of all the locally running AVM processes that is displayed by the debug launch configuration type. When you attach the Debugger to a local AVM, Progress Developer Studio for OpenEdge uses its process ID and attempts to make the process debug-ready on the specified port, if the AVM is not already running in the debug mode.

To make a successful connection with Progress Developer Studio for OpenEdge, you must start the external AVM in the debug mode on the specified host and port. Only one debug client can connect to the specified AVM at any point of time. Once attached, an entry for that AVM is displayed in the Debug view. If Progress Developer Studio for OpenEdge fails to connect to an external AVM, the Debug view shows the entry as terminated and displays an appropriate error message indicating connection failure.

The attached AVM notifies Progress Developer Studio for OpenEdge only if a breakpoint is reached. So, in order to debug a file on the attached AVM, you must set a breakpoint in the source file before it is executed by the attached AVM (See Setting breakpoints on page 573 for more information). You can add or remove breakpoints during debugging.

Progress Developer Studio for OpenEdge also enables you to specify the source lookup paths on which it searches for the source code of the program being executed by the attached AVM. For more information, see Edit Source Lookup Path dialog on page 624.

You can suspend or disconnect the Debugger from the attached AVM using the options available on the Debug view toolbar or the context menu. For more information, see Suspending and disconnecting the Debugger from an attached AVM on page 588.
See also

Remote Debugging on page 557
Attaching Debugger External AVM on page 572
Suspending and Disconnecting the Debugger from an Attached AVM on page 588
Progress External OpenEdge AVM Settings on page 615

AppServer and WebSpeed debugging

Progress Developer Studio for OpenEdge provides debugging support for remote and local AppServer and WebSpeed instances. You can use the Debugger to debug multiple AppServer or WebSpeed agents in a single Progress Developer Studio session.

You can attach the Debugger to an AppServer or WebSpeed broker by specifying the debug port (See Attaching the Debugger to an external AVM on page 572). Thus, you can debug the code running on the AppServer or WebSpeed broker. The broker provides a password as an optional parameter to ensure that it accepts debug connections only from authenticated clients.

Note: To debug an AppServer version prior to 11.0, you must use the standalone OpenEdge Debugger.

Progress Developer Studio for OpenEdge debug launch configuration type allows you to specify how to connect to the AppServer or WebSpeed server running in the debug mode (See Progress OpenEdge AppServer and WebSpeed settings on page 604). Upon a successful connection with the broker, you can view the broker and the agents running on it in the Debug view.

The Progress Developer Studio for OpenEdge Debugger also allows you to set breakpoints in an AppServer or WebSpeed debug session (See Setting breakpoints on page 573). When you set a breakpoint on a line of ABL code, the Debugger adds the breakpoint to all the AppServer or WebSpeed agents. When an agent reaches the breakpoint, it notifies the Debugger of its state through the broker. Once it receives the notification, Progress Developer Studio for OpenEdge verifies if the source file and the code being executed are the same. If the file in the Progress Developer Studio for OpenEdge environment is same as that in the AppServer or WebSpeed debug session, it opens in the ABL Editor and highlights the line on which the breakpoint was hit. The Debug view updates itself to display the agent on which the file is being executed.

The AppServer or WebSpeed debug session represents multiple AVMs, one for each AppServer or WebSpeed agent. You can view the variable information in the Variables view by selecting an individual agent node or one of the stack frames under it in the Debug view.

Note: The Debug view displays multiple AVMs in a debug session. Each agent represents a separate AVM and has a separate stack frame.

The Debugger also provides options to suspend and resume individual agents within an AppServer or WebSpeed debug session. For more information, see Suspending and resuming AppServer or WebSpeed agents on page 588.

See also

Remote Debugging on page 557
Attaching the Debugger External AVM on page 572
Setting Breakpoints on page 573
Suspending and Resuming AppServer or WebSpeed Agents on page 588
Progress OpenEdge AppServer and WebSpeed Settings on page 604
External OpenEdge Application Settings on page 615
**AppServer pass-through**

While debugging a client application, you can use an AppServer pass-through to step inside the remote procedure on the AppServer from within the client application. You can debug a remote procedure from within Progress Developer Studio for OpenEdge, during pass-through, by attaching Progress Developer Studio for OpenEdge to the respective broker or agent (See Attaching the Debugger to an external AVM on page 572). However, unlike in other debugging use cases, you need not set breakpoints in the remote procedure source code.

While attaching Debugger to the respective broker or agent, you must specify the source code of the remote procedure in the source lookup paths. For more information, see Source tab (Progress Source tabOpenEdge AppServer and WebSpeed) on page 612.

**See also**
Remote Debugging on page 557  
Code Stepping on page 560  
Attaching the Debugger to an External AVM on page 572

**SpeedScript debugging**

Progress Developer Studio for OpenEdge allows you to edit SpeedScript files. Embedded SpeedScript files are HTML files that contain SpeedScript (a subset of ABL) code contained within HTML SCRIPT elements. The SpeedScript editor automatically generates `.w` files from the HTML files in the SpeedScriptGen folder.

You can debug the SpeedScript code by setting breakpoints in the `.w` files. You can do this in two ways:

- Set the breakpoint on the Generate ABL tab where the `.w` file is generated
- Open `.w` the file using the ABL Editor and set breakpoints in it

For more information about SpeedScript files and debugging them, see the .

**See also**
Remote Debugging on page 557

**Debugging operations**

The Debugger provides several ways to control the execution of a program, analyze the results as the program runs, and make both persistent and non-persistent changes to the code while the debugging session is active.

**Options for controlling program execution**

The key to debugging is the ability to run a program and suspend execution at strategic points so that you can monitor and evaluate the results. To allow you to control the program flow in this manner, the Debugger includes the following features:

- **Breakpoints** - You can insert breakpoints on executable statements anywhere in your source code and in include files that might appear in many different procedures. The Debugger suspends execution at each breakpoint.
- **Code-stepping** - You can discretely execute the next statement, the next statement plus any sub-procedure or trigger called by that statement, or the remainder of the current procedure.
- **Suspend-Resume-Terminate** commands - You can explicitly interrupt or resume execution. You can also suspend and resume an attached external AVM at any time.
These features are more fully described in the following topics.

**Breakpoints**

Breakpoints are markers that you insert in the source code, or instructions that you define, to stop your application at certain points when it runs in a Debugger session. Used in conjunction with **stepping commands**, breakpoints let you fully control when the application runs uninterrupted and when it is suspended.

When the Debugger encounters a breakpoint, it suspends execution and takes control from the application. You can then examine the code as you wish, and continue when you are ready, either by stepping or by resuming and letting the application run until it reaches the next breakpoint.

A breakpoint can be conditional, where evaluation of a logical expression determines whether the interruption occurs. A breakpoint can be based on encountering an error, in which case the Debugger gets control at the next executable line after the line that caused the error.

All breakpoints defined in your workspace are known to the Debugger session. Without deleting any definitions, you can selectively enable only the ones that you want to use. Those breakpoints that you do not enable are ignored.

**Types of breakpoints**

You can set three different types of breakpoints:

- **At line** - Suspends execution at a specific point in the code. Optionally, you can define a condition that must be satisfied in order for the breakpoint to be honored.
- **On error** - Suspends execution when the program encounters an error condition. You can make the breakpoint apply to all errors, only to unsuppressed errors, or only to a single error condition that you specify.
- **Watchpoint** - Suspends execution when the value of a given variable or parameter changes. Optionally, you can define a condition that must be satisfied in order for the breakpoint to be honored.

**Conditional breakpoints**

For a conditional "at-line" or "watchpoint" breakpoint, the condition can be any expression that OpenEdge can evaluate to a logical value. The expression can include variables, buffer fields, object methods and attributes, logical operators (such as AND, OR), comparison operators (such as =, <>, EQ, NE), and other code entities.

OpenEdge validates the expression syntax when the breakpoint is added, but does not check variable references. If any variables in the condition are out of scope or nonexistent when the breakpoint is encountered, the Debugger suspends execution as if honoring the breakpoint and displays an error message.

**See also**

- Code-stepping on page 560
- Using breakpoints on page 573
- Breakpoints view on page 627

**Code-stepping**

Code-stepping refers to a set of commands that let you run procedures incrementally, suspending the process as each statement or block is executed so that you can evaluate the precise effect of each code segment. Used in conjunction with **breakpoints**, code-stepping commands let you fully control when the application runs uninterrupted and when it is suspended.

**Code-stepping options**

The following options are available:
• **Step Into** - Causes the Debugger to execute the current line and continue until it reaches the next executable statement, which may be in the current procedure, a subprocedure, or a trigger. That statement becomes the current line, and is not executed until you continue.

• **Step Over** - Causes the Debugger to execute the current line and continue until it reaches either a breakpoint, or the next executable statement in the same procedure. Thus, if the current line calls a subprocedure or causes a trigger to execute, the subprocedure or trigger is executed without interruption unless there is a breakpoint. In other words, the Debugger steps "over," not "into," the subprocedure or trigger.

• **Step Return** - Causes the Debugger to execute the current line and continue either until it reaches a breakpoint, or until the current procedure ends and returns control to the calling procedure.

• **Resume** - Causes the Debugger to execute the current line and continue without interruption until it reaches either a breakpoint or the end of the program.

• **Suspend** - Interrupts execution and gives control to the Debugger. Also, suspends an attached AVM.

• **Terminate** - Interrupts execution and ends the Debugger session.

• **Disconnect** - Disconnects the Debugger from the attached AVM.

**Garbage collection in Debugger**

The OpenEdge client uses a feature known as garbage collection. Garbage collection refers to the automatic deletion of ABL objects that have no current reference. Since the objects are no longer being used, removing them frees up resources and relieves you of having to explicitly delete each object created during a session.

With regard to the Debugger, garbage collection proceeds as follows:

• If an object is garbage-collected at the end of a statement, the debugger does not step into the destructor for the object (if the object has a destructor) unless there is a breakpoint in the destructor.

• The DELETE OBJECT statement steps into the destructor whether or not it contains a breakpoint.

• When stepping over a statement (as opposed to stepping into a statement), the Debugger stops in the destructor only if it contains a breakpoint, whether the object is deleted due to garbage collection or as the result of executing the DELETE OBJECT statement.

**See also**

- Breakpoints on page 560
- Stepping through the code on page 577
- Using the Debug view and stack trace on page 578
- Debug view on page 624

**Options for monitoring results**

During a debugging session, you can use the following views to keep track of current values:

• **Variables view** - Shows the values of all buffers, datasets, parameters, properties, variables, and temp-tables. Class variables, properties of a class type, arrays, buffers, temp-tables, datasets, and handle variables are all expandable in the view.

• **Expressions view** - Shows the values of data elements that you explicitly choose to display, providing a convenient way to monitor a specific set of values that interest you. Class variables, properties of a class type, arrays, buffers, temp-tables, datasets, and handle variables are all expandable in the view.

The Expressions view duplicates and also expands upon information available in the Variables view. You can use the Expressions view to show ABL data (for example, ACTIVE-WINDOW or SESSION) that is not available in the Variables view. (You can also see this level of detail in the Variable Details view.)
• **Dynamic Objects view** - Shows information about dynamically created database objects, user interface widgets, XML objects, and other objects that are active at the current state of program execution.

**See also**
The OpenEdge Debugger Perspective on page 555
Monitoring program execution on page 578
Debugger views on page 624

**Code changes while debugging**
During a debugging session, you may want to make changes to your source code.

You have full access to all **ABL Editor** functions to make persistent changes. These changes do not take effect in your current debugging session; you must recompile and start a new session.

In addition, you can make non-persistent changes, not saved in your source code, to the values of variables and parameters. These changes take effect immediately, without recompiling.

**See also**
Options for monitoring results on page 561
Changing program code on page 586
Set Value dialog on page 621
Variables View on page 626

**Tasks**

**Using launch configurations**

You can run and debug your ABL programs and AppServer or WebSpeed instances from within the Progress Developer Studio for OpenEdge environment. Each time you do so, the program runs a launch configuration, which you can customize to suit the needs of your application and its target implementation.

**See also**
Launch Configurations for Running and Debugging Programs on page 551
Launch Configuration Settings on page 589

**Running or debugging a program**

To run or debug an ABL program (.p, .w, or .cls file), an AppServer instance, or a WebSpeed instance that exists in a Progress Developer Studio for OpenEdge project, you need to decide on a launch configuration to use. You have the following options:

• Let OpenEdge launch the program automatically.

• Select and run an existing launch configuration.

• Explicitly define and run a new launch configuration.
Letting OpenEdge launch the program automatically

When choosing this technique, OpenEdge first tries to find an appropriate existing launch configuration. If no suitable configuration is found, a new one is created specifically for the selected program, using default settings.

1. Select the file in the Project Explorer view, or open it in the ABL Editor.

   **Note:** You can also launch a debugging session with focus on an include file.

2. Do one of the following:

   - Click Run or Debug on the main toolbar. Simply clicking the button image is normally sufficient to launch the program. You can also click the down arrow to open the drop-down menu, and select Run As or Debug As.
   - Right-click in the ABL Editor window and select Run As from the context menu.

The program runs in the configuration chosen or created by OpenEdge. Consider the pros and cons of this approach:

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. This is the quickest way to run a program.</td>
<td>1. The configuration that OpenEdge chooses or creates may not meet all of your needs.</td>
</tr>
<tr>
<td>2. You can use this technique to create a basic configuration, and then fine-tune the configuration by editing it in the Configurations wizard.</td>
<td>2. Using this technique repeatedly might create a large number of configuration definitions that you do not want to keep, resulting in the need to do some &quot;housekeeping.&quot; You can use the Configurations wizard to delete unwanted configurations.</td>
</tr>
<tr>
<td>3. After using the Configurations wizard to set up a configuration that is appropriate for most programs in a specific project, you can use this technique to run any program in the project under that configuration.</td>
<td></td>
</tr>
</tbody>
</table>

See also
Defining and running a new launch configuration on page 564
Selecting an existing launch configuration on page 564
**Selecting an existing launch configuration**

With this technique, you choose a launch configuration that you have previously defined.

1. Select the file in the **Project Explorer** view, or open it in the ABL Editor.

2. Click the down arrow on the Run button or the Debug button on the main toolbar to open the drop-down menu.

3. The configuration you want may appear on the top-level menu in the **Favorites** section, the recent history list, or both. If so, select it from that menu.

4. If the configuration is not offered on the top-level menu, select **Run Configurations** or **Debug Configurations** to open the **Configurations** wizard.

5. In the wizard, select the desired configuration from the tree view in the left pane and then click the Run or Debug button in the lower right corner.

The program runs in the selected configuration. If you used the Debug command, the Debugger starts; see **Launching the Debugger** on page 571 for information on how to proceed with the debugging session.

**See also**

* Defining and running a new launch configuration on page 564
* Letting OpenEdge launch the program automatically on page 563

**Defining and running a new launch configuration**

For complete control of the configuration definition, use the **Configurations** wizard.

1. Click the down arrow on the Run button or the Debug button on the main toolbar to open the drop-down menu.

2. Select **Run Configurations** or **Debug Configurations** to open the **Configurations** wizard.

3. Define the configuration. You can add it to the Run and Debug menus if you wish.

4. Run the configuration either from the Configurations wizard or from the Run or Debug menu.

**See also**

* Selecting an existing launch configuration on page 564
* Letting OpenEdge launch the program automatically on page 563

**Starting an AppServer or a WebSpeed instance**

To start an AppServer or a WebSpeed instance, first select the instance in the Servers view, which is part of the default OpenEdge Server perspective. To display this view if it is not open, select Window > Show View > Servers.

You have the following options:

- Let OpenEdge start the server automatically.
- Select and run an existing AppServer or WebSpeed launch configuration.
- Explicitly define and run a new AppServer or WebSpeed launch configuration.
Note: You can use the Progress Developer Studio for OpenEdge Debugger to debug an AppServer or WebSpeed instance. For more information, see AppServer and WebSpeed debugging on page 558. To debug an AppServer version prior to 11.0, you will need to use the standalone OpenEdge Debugger.

See also
AppServer and WebSpeed Debugging on page 558
Launch configurations for running and debugging programs on page 551
Managing launch configurations on page 566
Adding a configuration to the menu on page 568
Progress OpenEdge AppServer and WebSpeed settings on page 604

Letting OpenEdge start the AppServer or WebSpeed broker automatically
With this technique, OpenEdge first tries to find an existing launch configuration for the server. If none is found, a new one is created specifically for the selected server, using default settings.

- Click Start/Run or Debug on either the Servers view toolbar or the main toolbar. The AppServer or WebSpeed agent starts in the configuration chosen or created by OpenEdge.

See also
Defining and running a new launch or debug configuration on page 565
Selecting an existing AppServer or WebSpeed launch configuration on page 565

Selecting an existing AppServer or WebSpeed launch configuration
With this technique, you choose a launch configuration that was previously defined.

1. Click the down arrow on the Run button or the Debug button on the main toolbar to open the drop-down menu.
2. The configuration you want may appear on the top-level menu in the Favorites section, the recent history list, or both. If so, select it from that menu.
3. If the configuration is not offered on the top-level menu, select Run Configurations or Debug Configurations to open the Configurations wizard. In the wizard, select the desired configuration from the tree view in the left pane and then click the Run button or the Debug button in the lower right corner.

   The AppServer or WebSpeed agent starts in the selected configuration.

See also
Defining and running a new launch or debug configuration on page 565
Letting OpenEdge start the AppServer or WebSpeed broker automatically on page 565

Defining and running a new launch or debug configuration
For complete control of the configuration definition, use the Configurations wizard.
1. Click the down arrow on the **Run** button or the **Debug** button on the main toolbar to open the drop-down menu.

2. Select **Run Configurations** or **Debug Configurations** to open the **Configurations** wizard.

3. Define the configuration. You can add it to the **Run** and **Debug** menus if you wish.

4. Run the configuration either from the Configurations wizard or from the **Run** or **Debug** menu.

**See also**
- Selecting an existing AppServer or WebSpeed launch configuration on page 565
- Letting OpenEdge start the AppServer or WebSpeed broker automatically on page 565

### Managing launch configurations

You use a launch configuration every time you run or debug an ABL program, an AppServer or a WebSpeed instance in Progress Developer Studio for OpenEdge. The following topics in this section provide instructions for working with launch configurations.

#### Defining a launch configuration

Use the **Configurations** wizard to set all desired options for a new launch configuration or to edit an existing configuration.

1. To open the **Configurations** wizard, click the down arrow on either the **Run** button or the **Debug** button on the main toolbar to open the drop-down menu. Then select **Run Configurations** or **Debug Configurations**. Alternatively, right-click in the **ABL Editor** and select **Run As > Run Configurations** or **Debug As > Debug Configurations**.

   **Note:** For the purpose of defining a configuration, it makes no difference whether you use a **Run** command or a **Debug** command to open the wizard. In both cases, the wizard gives you access to all existing configuration definitions, and the process of creating or editing a configuration is identical.

2. Select the desired launch configuration type from the tree view in the left pane. The launch configuration types are: **Progress OpenEdge Application**, **Progress OpenEdge AppServer**, **Progress OpenEdge WebSpeed**, and **Progress External OpenEdge Application** (for Debug Configurations only).

3. To create a new launch configuration, click on the wizard toolbar and enter a name for the configuration in the **Name** field.

4. To edit an existing configuration, select it in the tree view in the left pane. The wizard includes tabs labeled **Main**, **Startup**, **PROPATH**, **Databases**, **ABL**, **Culture**, **Performance**, **Logging**, **Environment**, **Common**, **Server**, **Source**, **Security**, and **Connection**. Set values as necessary for options on all tabs. In many cases you need to specify only a few settings, as the default values are appropriate. These defaults are generally copied from settings for the project or the operating system, or from the `startup.pf` file in your OpenEdge root directory.

5. After setting the desired options, click **Apply** to save the configuration. You also have the option of using the configuration immediately by clicking **Run** or **Debug**.

A few things to keep in mind. Be aware of the following considerations:

- **Project selection** - You are required to associate the launch configuration with a project (the first field on the **Main** tab). Doing so does not limit you to using the configuration only in that project. However, your
choice affects many of the default values for the configuration. It is also important if you are creating a standard configuration to run whenever you select a program in that project and use the Run/Debug as Progress OpenEdge Application command.

- **Debug configurations** - The Progress Developer Studio for OpenEdge Debugger requires that the target program run under a dedicated instance of the AVM, not under the project AVM. When creating a configuration for debugging, be sure to specify Start new OpenEdge AVM on the Main tab.

- **Class files** - Similarly, ABL class files must run under a dedicated instance of the AVM, not under the project AVM. When creating a configuration for running classes, be sure to specify Start new OpenEdge AVM on the Main tab.

**See also**
- Launch configurations for running and debugging programs on page 551
- Running or debugging a program on page 562
- Creating project standard and program specific configurations on page 567
- Adding a configuration to the menu on page 568
- Copying and deleting launch configurations on page 568
- Setting launch configuration preferences on page 569
- Launch configuration settings on page 589

**Creating project-standard and program-specific configurations**

The quickest way to run or debug a program is to use the Run/Debug as command, or simply to click Run or Debug on the main toolbar while the program is selected. When you use one of these commands, OpenEdge first looks for a suitable existing launch configuration to use. If none is found, it creates a new one.

For the convenience of running different programs in a project with just a click or two and without the inconvenience of accumulating numerous automatically created configurations that you do not need, it is helpful to understand how OpenEdge chooses an existing configuration. You can take advantage of the way this selection process works to define:

- A project-standard configuration that will be used in a given project in all cases where no program-specific configuration exists.
- A program-specific configuration that will supersede the project-standard one if the program is selected when you issue the Run or Debug command.

**Defining a project-standard configuration**

To define a project-standard configuration, observe the following guidelines when setting values on the Main tab of the Configurations wizard:

- **Project setting** - Specify the project in which you want to use the configuration.
- **Startup program setting** - Choose Run selected program in workspace.

**Defining a program-specific configuration**

Using a standard launch configuration does not prevent you from using as many program-specific launch configurations as you need. When defining such a configuration, observe the following guidelines for values on the Main tab of the Configurations wizard:

- **Project setting** - Specify the project that contains the program.
- **Startup program setting** - Choose Run this program and specify the appropriate startup program.
When you select the program in the workspace and use the **Run/Debug as** command, OpenEdge chooses the program-specific configuration over any others that exist. You can also run the specified program from any project by explicitly choosing the associated launch configuration.

**See also**
- Launch configurations for running and debugging programs on page 551
- Running or debugging a program on page 562
- Defining a launch configuration on page 566
- Adding a configuration to the menu on page 568
- Copying and deleting launch configurations on page 568
- Setting launch configuration preferences on page 569
- Launch configuration settings on page 589

**Adding a configuration to the menu**

For launch configurations that you use frequently, it is helpful to add them to the top-level **Main** or **Debug** menu. These menus have a **Favorites** section (not visible when empty) at the top to which you can add any of your launch configurations. Use one of the following methods:

- **Configurations wizard** - Open the configuration definition in the wizard. On the Common tab, check **Run**, **Debug**, or both in the **Display in favorites menu** section.

- **Organize Favorites dialog** - Click the down arrow on either the **Run** button or the **Debug** button on the main toolbar to open the drop-down menu. Then select **Organize Favorites** and use the command buttons in the dialog to add, remove, or reorganize menu entries.

**See also**
- Launch configurations for running and debugging programs on page 551
- Running or debugging a program on page 562
- Configurations wizard on page 589

**Copying and deleting launch configurations**

The **Configurations** wizard lets you copy and delete existing configuration definitions. To open the wizard, click the down arrow on either the **Run** button or the **Debug** button on the main toolbar to open the drop-down menu. Then select **Run Configurations** or **Debug Configurations**.

- **To copy a configuration** - Select the configuration in the left pane of the wizard and click on the wizard toolbar. The wizard creates and opens a copy of the selected configuration. The duplicate configuration has the same name as the source, with an incremental number in parentheses appended.

- **To delete a configuration** - Select the configuration in the left pane of the wizard and click on the wizard toolbar, or press **Delete**.

**See also**
- Launch configurations for running and debugging programs on page 551
- Defining a launch configuration on page 566
- Creating project standard and program specific configurations on page 567
- Adding a configuration to the menu on page 568
- Setting launch configuration preferences on page 569
Setting launch configuration preferences

Both Progress Developer Studio for OpenEdge and the Eclipse framework offer options that affect launch configurations.

- To set Progress Developer Studio for OpenEdge launching preferences, select Window > Preferences > Progress OpenEdge > Advanced > Launching.

  You can choose the Configurations wizard defaults for two important settings on the Main tab:

  - **AVM usage** - By default, a new configuration runs under a separate, dedicated instance of the AVM. To change the default behavior so that new configurations use the same instance of the AVM as the project, check Use project AVM for new launch configurations. (In Debug mode, however, programs always run in a separate instance of the AVM, irrespective of the preference setting or the setting in the Configurations wizard.)

  - **Startup program** - By default, a new configuration definition identifies a specific program to be run. To change the default behavior so that new configurations run whatever program is currently selected, check Use selection in workspace for new launch configurations.

Eclipse launching preferences

To adjust various aspects of how the launching mechanism works, select Window > Preferences > Run/Debug > Launching.

See also

- Launch configurations for running and debugging programs on page 551
- Defining a launch configuration on page 566
- Creating project standard and program specific configurations on page 567
- Main tab Application on page 589

Using the Progress Developer Studio for OpenEdge Debugger

Setting Debugger preferences

To help you better manage your debugger environment, you can set two preferences specific to the operation of the Debugger.

1. Select Window > Preferences > Progress OpenEdge > Debug. The Debug Preferences dialog appears.

2. Optionally, set the following preferences:

   - **Enable property evaluation** - Check this option if you want the evaluation of properties with accessors to occur. This option is not enabled by default. If you do not select this option, you see Property evaluation disabled in the Variables view as the property value, instead of the actual value of the property. (A property added to the Expressions view is always evaluated, whether or not you select this option.)

   - **Always select first matching file upon CRC mismatch** - Select this option if you want to always select the first matching file and ignore the rest, if there are one or more files with similar name but none of them match with the CRC of the program being debugged.
Connection timeout (ms) - Provide the time-out value that you want the Progress Developer Studio for OpenEdge Debugger to use when connecting to an OpenEdge client. This option is enabled with a default value of 20000 milliseconds.

3. Click **Apply** to save any preference changes, or click **Restore Defaults** to return to the original preference setting(s).

4. Click **OK**.

**See also**

OpenEdge Debugger preferences on page 556
Debug preferences dialog on page 619

**Enabling debugging**

Debugging is automatically enabled for you in Progress Developer Studio for OpenEdge. If debugging has been disabled, however, you must enable it in one of the following ways:

- By choosing to enable debugging in your first Debugger session. Simply choose Yes at the prompt to enable debugging for all OpenEdge clients.
- By running the proDebugEnable command-line utility.
- By setting the `ENABLE_OPENEDGE_DEBUGGER` environment variable.

To run the `proDebugEnable` command-line utility:

1. On the **Windows Start > Programs** menu, select **Progress > OpenEdge > Proenv**. A command shell window opens and displays information about your OpenEdge installation, followed by a command prompt.

2. Enter the following command:

   `prodebugenable -enable-all`

   A confirmation message indicates that debugging is enabled. You set the `ENABLE_OPENEDGE_DEBUGGER` environment variable at the operating system level. When you do so, be sure to set the value of the variable as either **yes** or **true**. For more information about setting environment variables, consult the documentation for your operating system.

**Disabling debugging**

Debugging remains enabled unless you explicitly disable it. If you did not set the `ENABLE_OPENEDGE_DEBUGGER` environment variable, you can disable debugging simply by entering the following command from the Proenv window:

   `prodebugenable -disable-all`

   A confirmation message indicates that debugging is disabled. If you did set the `ENABLE_OPENEDGE_DEBUGGER` environment variable, you must also remove the variable.

**See also**

Prerequisites for using the Debugger on page 556
Launching the Debugger on page 571
Launching the Debugger

You launch the Debugger by selecting the desired source file and issuing the `Debug` command as explained in Running or debugging a program on page 562.

**Note:** You can also launch a debugging session with focus on an include file.

When you launch the Debugger, you may see either or both of two prompts:

- If you have not enabled debugging, you are prompted to do so before the debug session runs.
- If you are not in the Debugger perspective, you may be prompted to confirm a switch to that perspective. Switching is recommended but not required.

A dedicated OpenEdge runtime instance starts for the session in a suspended state, with an ABL stack trace displayed as a tree in the Debug view. The current line, which is the first executable statement in the source file, is selected in the stack trace. In an ABL Editor window showing the source file, this line is highlighted and includes an execution pointer (a) in the left margin.

Use any of the program execution commands on the Debug view toolbar or context (right-click) menu to proceed.

**Debugging class files**

When you use an ABL class file as the starting program for the Debugger, a temporary program is created to instantiate the class, using a public default constructor (no parameters). A different temporary program is used for each debugging session, as concurrent debug sessions are allowed. Each temporary program has a unique name relative to the Debugger session. If the class does not have a public default constructor, an error occurs.

**See also**

Options for controlling program execution on page 559
Launch configurations for running and debugging programs on page 551
Enabling debugging on page 570
Configuring for r-code generation on page 571
Controlling program flow on page 573
Monitoring program execution on page 578
Closing a debugger session on page 587
Defining a launch configuration on page 566
Copying and deleting launch configurations on page 568
Setting launch configuration preferences on page 569
Managing launch configurations on page 566
Running or debugging a program on page 562
Debugger dialogs on page 619
Debugger views on page 624
Launch configuration settings on page 589

**Configuring r-code generation**

The Debugger operates on compiled r-code. Therefore, you must specify the appropriate build options in your project properties to save .r files when you save or compile your code.
1. In the Project Explorer view, select your project and open its Properties window (right-click and select Properties, or select File > Properties from the menu bar).

2. Select Progress OpenEdge > Build, and make sure the Save r-code option is checked (the default setting). Set other values as appropriate.

Note: The location that you specify for .r files must be included in the PROPATH you specified when you created the launch configuration.

For more detailed instructions, see Build properties page or Defining a launch configuration on page 566.

See also
Prerequisites for using the Debugger on page 556
Defining a launch configuration on page 566

Attaching the Debugger to an external AVM

You can use the Progress Developer Studio for OpenEdge Debugger in attachable mode by attaching it to a debug-ready external AVM. The external AVM could be any one of the following:

- An AVM running outside Progress Developer Studio for OpenEdge on the same machine or a remote machine.
- An AppServer or WebSpeed agent running on the same machine or a remote machine.

To attach the Debugger to an external AVM:

1. Select a launch configuration under the External OpenEdge Application node on the Debug Configurations wizard. For more information on selecting a launch configuration, see Running or debugging a program on page 562.

2. On the Connections tab, select the connection type of the application to which you want to connect.

3. Select one of the following options:

   - Make debug-ready and attach (local host only) - Makes a process debug-ready and attaches the Debugger to the process. Select this option only if the external AVM is running on the same machine as the Debugger.
   - Attach to debug-ready process - Attaches the Debugger to a process that is debug-ready. Select this option if the external AVM instance is running on a remote machine.

4. On the Source tab, specify one or more source lookup paths on which Progress Developer Studio for OpenEdge searches for the source code of the program being executed by the attached AVM. For more information, see Source tab (Progress OpenEdge External AVM) on page 616.

Making a process debug-ready and attaching the Debugger to it

To make a process debug-ready and attach the Debugger to it:

1. Select Make debug-ready and Attach (local host only).

2. In the Process ID field, enter the process ID of the process on the local machine you want to debug. To list the currently running OpenEdge processes, click Select Process. The Select Process dialog box appears. For more information, see the Select Process dialog on page 623
3. In the **Port** field, enter a port number which you want to use for communicating with the client process, or leave the default value (0) for the AVM to determine a usable port number.

4. Click **OK**.

**Attaching the Debugger to a debug-ready process**

To attach the Debugger to a process that is already in a debug-ready state:

1. Select **Attach to debug-ready Process**.
2. Enter the name of the host on which the process is running, or accept the default localhost.

   **Note:** You must enter a **hostname** if the process is not running on the same machine as the Debugger.

3. Enter the **port number** used to make the process debug-ready.

4. Click **OK**.

When you select Debug on the debug launch configuration, Progress Developer Studio for OpenEdge tries to connect to the AVM specified by the host (or process ID) and the port number. Once connected, the Debug view displays the attached AVM.

   **Note:** The external AVM must be in the debug mode on the specified host and port in order to make a successful connection with Progress Developer Studio for OpenEdge. Only one debug client can connect to the specified AVM at a time.

**See also**

Attachable Debugger on page 557

Connection tab (Progress OpenEdge External AVM) on page 615

**Controlling program flow**

**Using breakpoints**

Breakpoints are markers that you insert in source code, or instructions that you define, to stop your application at certain points when it runs in a Debugger session.

**Setting breakpoints**

You can set three different types of breakpoints:

- **At Line** - suspends execution at a specific point in the code. Optionally, you can define a condition that must be satisfied in order for the breakpoint to be honored.

- **OnError** - suspends execution when the program encounters an error condition. You can make the breakpoint apply to all errors, only to unsuppressed errors, or only to a single error condition that you specify.

- **Watchpoint** - suspends execution when the value of a given variable, buffer field, object attribute, or object reference changes. Optionally, you can define a condition that must be satisfied in order for the breakpoint to be honored.
Breakpoints in ABL files outside workspace

When you use the Attachable Debugger on page 557, the source code for the program being debugged on the external AVM is located outside the Progress Developer Studio for OpenEdge workspace. To debug the file on the attached AVM, you must set a breakpoint in that file before it is executed by the AVM. You can add or remove breakpoints during debugging. The attached AVM notifies Progress Developer Studio for OpenEdge only if a breakpoint is reached.

**Note:** To set a breakpoint in the file outside the Progress Developer Studio for OpenEdge workspace, you must have access to its source code.

When Progress Developer Studio for OpenEdge receives a breakpoint hit notification from the attached AVM, it performs a source lookup for the notified file name. If it finds a matching file, it opens the file in the ABL Editor for debugging. If it does not find the file name, it notifies you accordingly and prompts you to edit the source lookup path. For more information, see Edit Source Lookup Path dialog on page 624.

**Note:** The breakpoints in files outside the workspace appear only in the Breakpoints view. The Breakpoint marker (a small blue circle (•) in the vertical rulers section on the left) does not appear for these breakpoints due to a limitation in Eclipse.

Breakpoints in an AppServer or a WebSpeed debug session

When you set a breakpoint on a line of ABL code while debugging an AppServer or a WebSpeed instance, the Debugger adds the breakpoint to all of the AppServer or WebSpeed agents. When any of these agents reaches the breakpoint, it notifies the debugger of its current state, through the broker. Upon receiving the notification, Progress Developer Studio for OpenEdge verifies if the source file and the code being executed are the same. If the file within the Progress Developer Studio for OpenEdge environment is the same as that in the AppServer or WebSpeed agent session, then the file opens up in the ABL Editor with the breakpoint hit line highlighted. The Debug view updates itself to display the agent on which the file is being executed.

Setting breakpoints programmatically

You can use the SET-BREAK and CANCEL-BREAK methods in ABL code to insert and remove breakpoints programmatically. However, the DEBUG and INITIATE methods, as well as the VISIBLE attribute, are ignored for Progress Developer Studio for OpenEdge Debugger sessions. This feature is helpful to control breakpoints at specific points in your code, regardless of how the surrounding code is modified, or to set breakpoints in include files.

Breakpoints and include files

Progress Developer Studio for OpenEdge allows you to set breakpoints in include files that might appear in many different procedures. During remote debugging, the external AVM notifies Progress Developer Studio for OpenEdge of a breakpoint hit in an include file while executing a program that contains the include file. You can also set breakpoints in include files in a program launched from within Progress Developer Studio for OpenEdge, while debugging a local AVM.

See also

Breakpoints on page 560
Attachable Debugger on page 557
Selectively enabling breakpoints on page 576
Managing breakpoints on page 576
Using watch expressions on page 583
Breakpoints view on page 627
New Breakpoint dialog on page 621
Expressions view on page 628
Edit Source Lookup Path dialog on page 624

Setting an at-line breakpoint

Setting an at-line breakpoint suspends execution at a specific point in the code. Optionally, you can define a condition that must be satisfied in order for the breakpoint to be honored. To set an at-line breakpoint:

1. To insert an unconditional breakpoint at a specific code line, you can simply double-click in the left margin next to that line in an ABL Editor window. Alternatively, with the cursor in the target line, right-click and select Toggle Breakpoint from the context menu. When you insert a breakpoint, a small blue circle (●) appears in the left margin.

2. To set a conditional breakpoint, or to set a breakpoint in a file that is not open for editing, use the New Breakpoint dialog as follows:
   a) Open the New Breakpoint dialog by right-clicking in the Breakpoints view and selecting Add Breakpoint.
   b) Select the At Line tab.
   c) Enter the name of a procedure file that can be found in the project PROPATH, or the name of an internal procedure found in the project. You can leave the Procedure Name field blank if the file to which you are adding the breakpoint is open in the currently selected Editor window.
   d) Enter the line number. You can leave the Line Number field blank to add the breakpoint at the first line of the specified procedure.
   e) Optionally, enter a condition statement. The Debugger will suspend execution at the breakpoint only if the condition is true at that time.

You can go quickly to the breakpoint location in an Editor window. Select an at-line breakpoint in the Breakpoints view, right-click, and select Go to File from the context menu.

See also
Setting a watchpoint on page 576
Setting an on-error breakpoint on page 575

Setting an on-error breakpoint

Setting an on-error breakpoint suspends execution when the program encounters an error condition. You can make the breakpoint apply to all errors, only to unsuppressed errors, or only to a single error condition that you specify. To set an on-error breakpoint:

1. Open the New Breakpoint dialog by right-clicking in the Breakpoints view and selecting Add Breakpoint.
2. Select the On Error tab.
3. Select one of the options:
   • On Any Unsuppressed Error - The breakpoint takes effect only if the error occurs on execution of a statement that does not have the NO-ERROR attribute.
   • On Any Error - The breakpoint takes effect on any error, suppressed or unsuppressed.
   • On Error - The breakpoint takes effect only on the specific error that you identify in the Error Number field.

See also
Setting a watchpoint on page 576
Setting an at-line breakpoint on page 575
Setting a watchpoint

Setting a watchpoint breakpoint suspends execution when the value of a given variable, buffer field, object attribute, or object reference changes. Optionally, you can define a condition that must be satisfied in order for the breakpoint to be honored. To set a watchpoint breakpoint:

1. Open the **New Breakpoint** dialog by right-clicking in the **Breakpoints** view and selecting **Add Breakpoint**.
2. Select the **Watchpoint** tab.
3. In the **Expression** field, enter the name of the data element whose value you want to track.
4. Select the **Add Watch** option if you want to show this expression in the **Expressions view**.

   Optionally, enter a **condition statement**. The Debugger will suspend execution only if the condition is true when the value changes for the specified expression.

**See also**

- Setting an on-error breakpoint on page 575
- Setting an at-line breakpoint on page 575

Selectively enabling breakpoints

1. The **Breakpoints** view lists all breakpoints that are defined for the current workspace and for the files located outside the workspace. Each entry in the list has a checkbox. You can disable any breakpoint by clearing its checkbox, or re-enable it by checking it.
2. The context menu for the **Breakpoints** view provides a **Select All** command and **Enable** and **Disable** commands, which are useful for operating on the multiple breakpoints. Right-click in the **Breakpoints** view and select the appropriate command.
3. As a shortcut for disabling all breakpoints, click **Skip All Breakpoints** on the **Breakpoints** view toolbar. To re-enable the skipped breakpoints, click this button again. (Those breakpoints that are unchecked in the list remain disabled.)

**See also**

- Breakpoints on page 560
- Setting breakpoints on page 573
- Managing breakpoints on page 576
- Breakpoints view on page 627

Managing breakpoints

The **Breakpoints** view lists all breakpoints defined for the current workspace and for the files located outside the workspace. You can edit or delete any breakpoint in the list.

- To edit breakpoints:
  a) You can change the definition of any breakpoint. However, you cannot change its type (**At Line**, **On Error**, or **Watchpoint**); nor can you change the procedure in which an at-line breakpoint is located.
  b) To edit a breakpoint, select it by clicking the text portion of the list entry. Then right-click and select **Edit Breakpoint** to display the **Edit Breakpoint** dialog. Change the values as appropriate and click OK. See **Setting breakpoints** on page 573 for more information.

- To delete breakpoints:
a) To delete a breakpoint, select it by clicking the text portion of the list entry. To delete more than one, select with Ctrl +click. Then press Delete, or select Remove Selected Breakpoints or right-click and select Remove.

b) The context menu for the Breakpoints view provides Select All, Remove, and Remove All commands, which are useful for operating on the whole list. Right-click in the Breakpoints view and select the appropriate command. Remove All is also available on the toolbar.

Note: You can temporarily disable breakpoints without deleting them. See Selectively enabling breakpoints.

- To share breakpoints, use the Export and Import options on the Breakpoints view context (right-click) menu to share breakpoint definitions with other users. See for more information.

See also
Breakpoints on page 560
Setting breakpoints on page 573
Breakpoints view on page 627

Stepping through the code
While a Debugger session is active, use the following commands on the Debug view toolbar or context (right-click) menu to execute the program incrementally:

- **Step Into** - Causes the Debugger to execute the current line and continue until it reaches the next executable statement, which may be in the current procedure, a subprocedure, or a trigger. That statement becomes the current line, and it is not executed until you continue.

- **Step Over** - Causes the Debugger to execute the current line and continue until it reaches either a breakpoint, or the next executable statement in the same procedure. Thus, if the current line calls a subprocedure or causes a trigger to execute, the subprocedure or trigger is executed without interruption unless there is a breakpoint. In other words, the Debugger steps “over,” not “into,” the subprocedure or trigger.

- **Step Return** - Causes the Debugger to execute the current line and continue either until it reaches a breakpoint, or until the current procedure ends, returning control to the calling procedure.

- **Resume** - Causes the Debugger to execute the current line and continue without interruption until it reaches either a breakpoint or the end of the program.

- **Suspend** - Interrupts execution following completion of the current instruction and gives control to the Debugger at the next executable statement. Also, suspends the attached AVM.

- **Terminate** - Interrupts execution and ends the Debugger session.

- **Disconnect** - Disconnects the Debugger from the attached (running) external AVM and stops receiving notifications from it.

Note: These commands are enabled only when a Debugger session is active and, with the exception of the Suspend command, only when an entry in its stack trace is selected.
Chapter 13: Running and debugging ABL programs

Garbage collection
For details about the behavior of code-stepping with regard to garbage collection, see Garbage collection in the Debugger.

See also
Code-stepping on page 560
Breakpoints on page 560
Using the Debug view and stack on page 578
Using breakpoints on page 573
Debug view on page 624

Monitoring program execution

Using the Debug view and stack trace
The Debug view contains the stack trace and the program execution commands.

Stack trace
For each currently active Debugger session (and any terminated sessions that you have not removed), the Debug view contains one or more session trees. As shown in the example below, a session tree begins with a node that identifies the project as an OpenEdge application and shows its name ("Killer App"). The tree ends with a node identifying the location of the OpenEdge Runtime executable, prowin32.exe.

Stack entries, such as the highlighted entry in the example, appear marked with under the OpenEdge Runtime node.

Note: Stack entries are visible only when the Debugger is suspended. While the application is running or waiting for a user response (for example, at a message dialog), you cannot see the stack.

When the Debugger is in a suspended state, the stack trace identifies:

• The current code line. This is the next line of code to be executed, and corresponds to the location of the solid pointer and the highlighted line in the Editor window. The current line appears at the top of the stack.

• All other code lines that participated in the currently active code block, leading to the pending execution of the current line. These entries appear, from top to bottom, in inverse order of execution. That is, the entry for the most recently executed instruction appears first, immediately following the current (pending) instruction.

All visible stack entries belong to a discrete block of code whose execution is not yet complete. Once the block is fully executed, its stack entries are removed, and the stack for the next block appears.
Using the stack for code analysis:
Selecting a stack entry lets you view that line of the source code and the values of variable data elements as they are at that point in the execution of the program. When you click on a stack entry:

- The ABL Editor window containing that source file becomes active; the selected line is color-highlighted ( or ), with an execution pointer ( or ) in the left margin. When you click the Editor tab label, the cursor is positioned at the beginning of that line. You can simply review the code or edit, save, and recompile it as necessary.
  - If the selected line is the current line (the first stack entry), the execution pointer is solid and the highlighting is a darker shade:.
  - If the selected line is a previous line (lower in the stack), the execution pointer is hollow and the highlighting is a paler shade:

- The Variables view and the Expressions view show the values that are in effect at the time immediately preceding execution of the selected line. You can make non-persistent changes to these values to test their effect without actually altering the source code.

Note: Selecting a previously executed stack entry does not change the point at which program execution will resume. The current code line, reflected by the entry at the top of the stack, remains the same.

Program execution commands
The Debug view toolbar and context (right-click) menu provide commands for controlling program execution during a debugging session. See Stepping through the code for information about using these commands.

See also
Stepping through the code on page 577
Changing program code on page 586
Launching the Debugger on page 571
Debug view on page 624

Using the Variables view
When program execution is suspended during a Debugger session, the Variables view displays information about buffers, datasets, parameters, properties, temp-tables, variables, class variables, handles, and properties of a class type. You can see as little or as much of the information as you want, and you can also choose its overall format display.

- To open the Variables view if it is not open select Window > Show View > Debug > Variables.

See also
Options for monitoring results on page 561
The OpenEdge Debugger perspective on page 555
Using the Debug view and stack trace on page 578
Using the Variable Details dialog on page 582
Changing program code on page 586
Using watch expressions on page 583
Variables view on page 626
Selecting Variables view display information

To select the information you want to see in the Variables view:

1. To open the Variables view if it is not open select Window > Show View > Debug > Variables.

2. Click the drop-down menu from the Variables view toolbar.

3. Choose Layout > Show Columns. (You cannot select specific columns until you select the Show Columns option.)


5. Select each individual column you want to see in the view, or click Select All to see all the columns.

The information shown reflects the state of the data elements at the point immediately preceding execution of the line that is currently selected in the stack trace. When there are multiple entries in the stack, you can see how the data elements change at different stages of execution by selecting different stack entries.

Note: Note that you can also see the information provided in the Variables view by hovering the mouse on a variable name in the Editor window when the debug session is selected in the Debug view.

Expanding and collapsing the Variables view list

Buffers, temp tables, datasets, class variables, arrays, and properties of a class type have subordinate elements (fields or tables), which in turn may have subordinate elements of their own (for example, an array index). These hierarchical elements appear in the Variables view with a plus symbol (+) before the name.

- To open the Variables view if it is not open select Window > Show View > Debug > Variables.
- To expand the hierarchy and show the subordinate values, click the plus symbol (+).
- To collapse the hierarchy selectively, clicking the minus sign symbol (-);
- To collapse all entries and show only top-level data elements, click the Collapse All on the toolbar.

Note: Any object reference or handle that has not been initialized cannot be expanded. Arrays, buffers, and temp-tables can be expanded, whether initialized or not.

Evaluating a property in the Variables view

To evaluate a property from the Variables view.

- To open the Variables view if it is not open select Window > Show View > Debug > Variables.
- Select the property, right-click, and select Evaluate Property from the context menu.

This is helpful if you opted not to select the Enable property evaluation option in the Debug preferences, and you now want to see the value of the property in the Variables view. (When the preference option is not enabled, you see "Property evaluation disabled" instead of the value.)

Filtering the Variables view display

You can filter the display in the Variables view to show only selected element types.
1. To open the Variables view if it is not open select Window > Show View > Debug > Variables.

2. Click on the toolbar and select Filter from the drop-down menu.

   The available options you can show are buffers, datasets, parameters, properties, variables, and temp-tables. This filter applies to all instances shown in the top level of the Variables view.

3. The submenu shows a check mark next to the options that are currently enabled for display. Click any item to toggle its display option off or on.

**Temporarily changing values in the Variables view**

You can make non-persistent changes to variable values to test their effect from the current execution point. The new values remain in effect for the duration of the current Debugging session, or until they change programmatically during the current session. To change an element's value:

1. To open the Variables view if it is not open select Window > Show View > Debug > Variables.
2. Click the current value in the grid to make it editable and enter the replacement value. Alternatively, right-click the element in the Variables view and select Change Value from the context menu to display the Set Variable dialog.
3. Enter the replacement value, and click OK.

   In values that you enter in the Variables view, the backslash (\) is evaluated as an escape character, which can affect program execution. To be sure of getting the intended results, it is recommended that you enter a double backslash (\\) in all cases to represent a single backslash.

   The value you supply must be valid for the declared data type and the current program context. If you enter an invalid value, an error message appears.

**Adding watch expressions using the Variables view**

You can add a watch expression for one or more elements so that you can track their values in the Expressions view.

To add a watch expression:

1. To open the Variables view if it is not open select Window > Show View > Debug > Variables.
2. Select the desired entries in the Variables view, right-click, and select Create Watch Expression from the context menu

**Assigning a value to a data member**

Since class variables are expandable in the Variables view, data members are shown. These data members can be assigned as if they were variables at the top level of the Variables view.

1. To open the Variables view if it is not open select Window > Show View > Debug > Variables.
2. Assign a value to a data member in any of the following ways:
   - By updating the Value column in the Variables view.
   - By updating the value in the Detail pane.
   - By using the Change Value option from the Variables view's context menu.
Opening the Variable Details dialog

The Variable Details dialog provides an alternative way to view information about a single data element at a time. It is particularly useful for examining HANDLE objects, because it includes information about attributes and fields.

1. To open the Variables view if it is not open select Window > Show View > Debug > Variables.
2. To open the Variables Detail dialog, you must have an active Debugger session and have data displayed in the Variables view. Choose one of the following methods to open the Variable Details dialog:
   - Right-click the Variables view entry that you want to look at, and select Variable Details from the context menu.
   - Select a Variables view entry, and then select Variable Details from the OpenEdge menu.

The dialog opens, showing the current details for the selected data element. Since it is a modal dialog, the Variable Details dialog does not provide dynamic updates while you step through the program as does the Variables view.

Information appears on one or more of four tabs: Attributes, Fields, Array, and Value. Click a tab to view its contents. On tabs that are not applicable for the selected data element, or for which the information is not currently available, a message indicates that the element is not valid for that tab.

Viewing variables in an AppServer or WebSpeed debug session

An AppServer or WebSpeed broker debug session represents multiple, discrete AVMs, one for each AppServer or WebSpeed agent. To display the variable information:

1. To open the Variables view if it is not open select Window > Show View > Debug > Variables.
2. Select an individual agent node or one of the stack frames under it in the Debug view.

   If you select a debug node with no variables in context, the Variables view does not display any information.

   In an AVM session, you must suspend the agent for the Variables view to display any information.

Using the Variable Details dialog

The Variable Details dialog displays detailed information about a single selected data element. It includes much of the information that is available in the Variables view.

- To open the Variables Detail dialog, you must have an active Debugger session and have data displayed in the Variables view. Choose one of the following methods to open the Variable Details dialog:
  - Right-click the Variables view entry that you want to look at, and select Variable Details from the context menu.
  - Select a Variables view entry, and then select Variable Details from the OpenEdge menu.

The dialog opens, showing the current details for the selected data element. Since it is a modal dialog, the Variable Details dialog does not provide dynamic updates while you step through the program as does the Variables view.

Information appears on one or more of four tabs: Attributes, Fields, Array, and Value. Click a tab to view its contents. On tabs that are not applicable for the selected data element, or for which the information is not currently available, a message indicates that the element is not valid for that tab.

Changing the element selection:
To view information for a different data element, enter the element name at the top of the window and click View. There are three ways to enter an element name:

- Type the name in the View field.
- For a subordinate element of the current selection, click the name in the browse.
- Click the down arrow at the right of the View field and select an entry from the drop-down list of recently used element names.

**Temporarily changing values**

You can make non-persistent changes to the values of variables, attributes (writable only), or fields to test the effect of the changes from the current execution point. The new values remain in effect for the duration of the current Debugging session, or until they change programatically during the current session.

To change an element's value, click the current value in the grid to make it editable, and enter the replacement value. The field is not editable if the data definition or the current program context does not allow the value to be modified.

The value you supply must be valid for the declared data type. If you enter an invalid value, an error message appears.

**Note:** You cannot make changes on the Value tab, only in the Value column on other tabs. To modify the entry that appears on the Value tab, close the Variable Details dialog and edit the value in place in the Variables view.

**Adding a watch expression**

You can add a watch expression for the selected element so that you can track its value in the Expressions view. Simply click Add Watch to add the expression.

**See also**

Options for monitoring results on page 561
Using the Variables view on page 579
Changing program code on page 586
Using watch expressions on page 583
Variable Details dialog on page 620

**Using watch expressions**

A watch expression, or watch, is the name of a data element or built-in function whose value appears in the Expressions view. You control the content of the Expressions view by setting watches on those elements that you want to monitor. The values are dynamically updated as you step through the code or as you select different stack trace entries.

You can expand class variables and properties of a class type, as well as arrays, buffers, temp-tables, datasets, and handle variables in the Expressions view. If an object reference or handle has not been initialized, however, you cannot expand it. Arrays, buffers, and temp-tables can be expanded even if they are not initialized.

The Debugger perspective provides several shortcuts for adding watch expressions:

- Check the Add Watch option when setting a watchpoint
- Select Create Watch Expression from the Variables view context menu.
- Click Add Watch in the Variable Details dialog.
In addition to using these shortcuts, you can right-click in the Expressions view, select Add Watch Expression, and enter an expression directly in the Add Watch Expression dialog. See Expression syntax below.

The Add Watch Expression dialog includes an Enable option, which is checked by default. Deselect this option if you want to add the watch but temporarily make it inactive.

Expression syntax:
Valid watch formats are any of the following:

- The name of a variable, field, or parameter
- `handle:attribute`
- `widget-name:attribute [IN FRAME frame-name]`
- `{ FRAME | MENU } widget-name:attribute`
- `{ SUB-MENU | MENU-ITEM } widget-name:attribute [IN { MENU | SUB-MENU } menu-name]`

You can chain attribute references. For example, the following attribute reference displays the value of the TITLE attribute for a window containing a fill-in (assuming the fill-in is three levels from the window):

```
FILL-IN-1:PARENT:PARENT:PARENT:TITLE:
```

In addition, the following built-in functions can be watch expressions:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_PCONTROL</td>
<td>FRAME-VALUE</td>
</tr>
<tr>
<td>_SERIAL-NUM</td>
<td>GO-PENDING</td>
</tr>
<tr>
<td>COM</td>
<td>LASTKEY</td>
</tr>
<tr>
<td>CURRENT-LANGUAGE</td>
<td>MACHINE-CLASS</td>
</tr>
<tr>
<td>DATASERVERS</td>
<td>MESSAGE-LINES</td>
</tr>
<tr>
<td>DBNAME</td>
<td>NUM-ALIASES</td>
</tr>
<tr>
<td>ETIME</td>
<td>NUM-DBS</td>
</tr>
<tr>
<td>FRAME-DB</td>
<td>OPSYS</td>
</tr>
<tr>
<td>FRAME-FIELD</td>
<td>OS-DRIVES</td>
</tr>
<tr>
<td>FRAME-INDEX</td>
<td>OS-ERROR</td>
</tr>
<tr>
<td>FRAME-NAME</td>
<td>PROGRESS</td>
</tr>
<tr>
<td></td>
<td>PROMSGS</td>
</tr>
<tr>
<td></td>
<td>PROPATH</td>
</tr>
<tr>
<td></td>
<td>PROVERSION</td>
</tr>
<tr>
<td></td>
<td>RETRY</td>
</tr>
<tr>
<td></td>
<td>RETURN-VALUE</td>
</tr>
<tr>
<td></td>
<td>SELF</td>
</tr>
<tr>
<td></td>
<td>TIME</td>
</tr>
<tr>
<td></td>
<td>TODAY</td>
</tr>
<tr>
<td></td>
<td>TRANSACTION</td>
</tr>
</tbody>
</table>

Managing the expression list:

The Expressions view context (right-click) menu includes several commands for modifying or controlling the display. Depending on the element in focus and its context menu options, the commands you see can include the following:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select All</td>
<td>Lets you select all the expressions in the Expressions view.</td>
</tr>
<tr>
<td>Copy Expressions</td>
<td>Lets you copy one or more expressions.</td>
</tr>
<tr>
<td>Remove and Remove All</td>
<td>Let you delete expressions.</td>
</tr>
<tr>
<td>Find</td>
<td>Lets you find an element.</td>
</tr>
</tbody>
</table>
### Tasks

<table>
<thead>
<tr>
<th>Change Value</th>
<th>Lets you enter a new value for an element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add Watch Expression</td>
<td>Lets you add a watch expression and optionally enable it.</td>
</tr>
<tr>
<td>Reevaluate Watch Expression</td>
<td>Lets you reevaluate an expression.</td>
</tr>
<tr>
<td>Disable and Enable</td>
<td>Let you control whether or not selected expressions are updated. You might choose to disable some watches to reduce the time needed to update the Expressions view, or to reduce screen clutter. Watches are enabled by default when you add them.</td>
</tr>
<tr>
<td>Edit Watch Expression</td>
<td>Lets you change the syntax of the selected expression. You can also toggle its enabled/disabled status.</td>
</tr>
<tr>
<td>Create Watch Expression</td>
<td>Lets you create a watch expression.</td>
</tr>
</tbody>
</table>

### See also

- Options for monitoring results on page 561
- Setting breakpoints on page 573
- Using the Variables view on page 579
- Expressions view on page 628

### Using the Dynamic Objects view

#### Demo

**Monitoring Dynamic Objects in the Debugger**

The Dynamic Objects view lets you monitor the creation and destruction of dynamic object instances in an ABL application. Use this dialog box to locate dynamic object instances the application has created, no longer needs, and has not yet destroyed. These undeleted object instances might indicate a memory leak.

You must explicitly start dynamic object monitoring if you want to use this view. You can stop and restart monitoring during a debugging session, but each new session begins with monitoring turned off. If you have multiple active debugging sessions, you turn monitoring on or off for each session independently. When you start monitoring, you choose the specific object types to monitor.

When monitoring is turned on, you can filter the Dynamic Objects view display to show a subset of the object types that are being monitored. The display includes objects of a given type only if you both enable monitoring of that type and include the type in your filter setting.

#### See also

- Options for monitoring results on page 561
- Debug view on page 624
- Dynamic Objects view on page 629
- Filter Objects dialog on page 623

### Starting and stopping dynamic object monitoring

Each new Debugger session that you launch begins with dynamic object monitoring turned off.
To start dynamic object monitoring:

1. Start the Debugger session and make sure the OpenEdge Runtime node, or one of its stack entries, is selected in the Debug view.

2. Select OpenEdge > Dynamic Object Monitoring on the menu bar to display the Dynamic Object Monitoring dialog.

   The tree view groups object types in four categories: Dynamic Objects, User Interface Widgets, XML Objects, and Other Objects.

3. Expand any of these nodes to see the individual object types in the category. Object types that are selected for monitoring are checked.

4. Click the check boxes to toggle the selections on or off. You can select individual object types, or you can operate on a whole category by clicking the check box for its top-level node.

5. Click Start Monitoring and then Close.

To change settings during the session, open the Dynamic Object Monitoring dialog again. You can modify the selection of object types to be monitored and use the Stop Monitoring and Restart Monitoring commands.

You can also use Restart Monitoring to clear the display in the Dynamic Objects view and continue monitoring.

Filtering the Dynamic Object view display

You can choose the object types that you want to see in the Dynamic Objects view:

1. Make sure that monitoring is turned on for the appropriate types, as explained in Starting and stopping dynamic object monitoring on page 585.

2. Right-click in the Dynamic Objects view and select Filter Objects from the context menu.

3. In the Filter Objects dialog, object types that are selected for display are checked. Click the check boxes to toggle the selections on or off, and click OK.

Changing program code

During a debugging session, you can modify the behavior of your application in two ways:

- You can edit your source code, using the standard ABL Editor. Changes do not take effect in the current debugging session unless the program subsequently calls the modified procedures.

- You can change variable values temporarily. These changes take effect in the current debugging session but are not saved to the source code.

Making persistent changes by editing the ABL source

During a session, the Debugger opens each source file it processes in an Editor window. You can edit the file in that window, using all standard features of the ABL Editor, without leaving the Debug perspective.

You may want to have access to the tools available in the OpenEdge Editor perspective, such as the Outline view and the DB Structure view, particularly for extensive or complex changes. You can switch to the OpenEdge Editor perspective or any other perspective, keeping your debugging session active, and then return to the Debug perspective to continue the session.

Changes take effect the next time your application executes the modified code after you recompile.

Temporarily changing variable values

During a running debugging session, you can experiment to see how changes to variable values affect program execution, without modifying the source code. The Variables view and the Variable Details dialog let you make non-persistent changes, as explained in the topics covering the use of those tools.
See also
Using the Variables view on page 579
Using the Variable Details dialog on page 582

Going to a specific source line

You can use the line number of a debug listing file to go directly to a specific source line in the currently active file in the OpenEdge ABL Editor. A debug listing file is a temporary file generated by the Debugger during a debug session. It contains a list of line numbers of a procedure with all the preprocessor directives expanded and all the include files inserted. The debug listing file is removed once the debug session ends.

To go to a specific source line number using the debug listing line number:

• Select **Go to Source Line** from the **Navigate** menu on the main menu bar and enter the debug listing line number of the corresponding file that is currently open in the editor. Alternatively, you can press **ALT+L** to enter the debug listing line number.

  The source file opens in the editor with the cursor positioned at the corresponding source line number. If the source file is not located in the file system, an error occurs and you will be prompted to select the file from a different location.

**Note:** The file that is currently open in the editor must have the same name as the file present in the ABL stack trace and it must contain r-code. If the file does not contain r-code, it is automatically compiled and the generated r-code file is removed once the operation is complete.

Closing a Debugger session

At any time during an active session that the Debugger has control (that is, program execution is suspended), you can end the session by clicking **Terminate** on the **Debug** view toolbar or context (right-click) menu. A session terminates automatically when the program completes execution.

To clear the session tree of a terminated session from the **Debug** view, click **Remove**.

The context menu includes several additional shortcuts for terminating, terminating and relaunching, and removing sessions.

See also
Using the Debug view and stack trace on page 578
Stepping through the code on page 577
Closing a remote debugging session on page 587

Closing a remote debugging session

To disconnect from a remote debugging session:

• Click **Disconnect** on the **Debug** view toolbar or the context (right-click) menu.

  Upon disconnecting, Progress Developer Studio for OpenEdge cleans up any Debugger communications with the broker and its agents, while they continue to run.
Suspending and disconnecting the Debugger from an attached AVM

- To suspend an attached AVM click **Suspend** on the **Debug** view toolbar or the context (right-click) menu.

  Suspending an AVM displays its stack trace in the **Debug** view. Upon suspending the AVM, Progress Developer Studio for OpenEdge looks for the source of the program (shown in the top stack frame) in the source lookup path. When it finds the source file, it opens it in the **ABL Editor** and starts debugging it.

- To disconnect the Debugger from the attached AVM, select **Disconnect** on the **Debug** view toolbar or the context (right-click) menu. Once you disconnect from the AVM, Progress Developer Studio for OpenEdge stops receiving any notifications from it. However, the AVM continues to run.

Suspending and resuming AppServer or WebSpeed agents

To suspend individual agents within an AppServer or WebSpeed debug session:

- Highlight the agent node in the **Debug** view’s tree view and select **Suspend**.

  If the agent is currently active and cannot be suspended (for instance, the agent might be waiting for a request), it resumes to its previous state, and Progress Developer Studio for OpenEdge updates the UI appropriately. If the agent can be suspended properly, then Progress Developer Studio for OpenEdge is notified (through the broker) of the agent’s current state and updates the UI, displaying the current position in the code.

See also

Attachable Debugger on page 557
Using the Debug view and stack trace on page 578
Stepping through the code on page 577
Closing a Debugger session on page 587
Suspending and disconnecting the Debugger from an attached AVM on page 588
Debug view on page 624
Reference

Launch configuration settings

The topics in this section describe the available settings and controls related to defining and using launch configurations.

Configurations wizard

The topics in this section describe the options and controls available for defining launch configurations in the Configurations wizard.

Progress OpenEdge Application settings

The topics in this section describe the options and controls available for defining Progress OpenEdge Application launch configurations in the Configurations wizard.

This wizard appears when you select Run Configurations or Debug Configurations from:

- The drop-down menu on either the Run button or the Debug button, or
- The ABL Editor context (right-click) menu

Although the window title includes either "Run" or "Debug," depending on how you open the wizard, in both cases you have access to all launch configurations, and the options are identical.

Various options have the effect of invoking a startup parameter for the AVM session. In such cases, the relevant parameter is shown in the rightmost column of the table. See OpenEdge Deployment: Startup Command and Parameter Reference, available at the Progress Product Documentation Web site, for detailed information.

There is a separate topic for each of the tabs in this section of the wizard.

Main tab (Application)

The Main tab in the Progress OpenEdge Application section of the Configurations wizard contains the required settings for defining an OpenEdge Application launch configuration, as well as some other frequently used options for starting an ABL session. See Progress OpenEdge Application settings on page 589 for information about accessing the wizard.

The following controls are available on the Main tab:

<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
<th>Related parameter (if applicable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name (accessible from all tabs)</td>
<td>(Required) Specifies the name of the launch configuration, which appears on menus and in the left pane of the Configurations wizard.</td>
<td></td>
</tr>
<tr>
<td>Project</td>
<td>(Required) Specifies the existing OpenEdge project in which the configuration is most often used. You can also use the configuration in other projects. See How OpenEdge creates or reuses a configuration on page 553 for information about how this selection affects launch configuration behavior.</td>
<td></td>
</tr>
<tr>
<td>Control</td>
<td>Purpose</td>
<td>Related parameter (if applicable)</td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>Run this program/Run selected program in workspace</td>
<td>Controls whether the configuration runs the program that currently has focus (in the Project Explorer view or in the ABL Editor), or the specific program identified in the Startup program field.</td>
<td></td>
</tr>
<tr>
<td>Startup program</td>
<td>(Required if Run this program is selected) Specifies the procedure that executes when the configuration runs.</td>
<td></td>
</tr>
<tr>
<td>Working directory</td>
<td>(Required) Specifies the directory in which the AVM starts when the configuration runs.</td>
<td></td>
</tr>
<tr>
<td>Use project AVM/Start new AVM</td>
<td>Controls whether the configuration runs under the instance of the AVM that is already running for the current project, or under a new, dedicated instance of the AVM.</td>
<td></td>
</tr>
<tr>
<td>OpenEdge version</td>
<td>Specifies the currently installed version of OpenEdge under which the configuration runs.</td>
<td></td>
</tr>
<tr>
<td>Use TTY</td>
<td>If checked, causes the configuration to run procedures in a character client window rather than a GUI window (that is, to run _progres.exe instead of _prowin32.exe).</td>
<td></td>
</tr>
<tr>
<td>Batch mode</td>
<td>If checked, causes the configuration to run procedures in a batch session, with no terminal interaction.</td>
<td>Batch (-b)</td>
</tr>
<tr>
<td>Automatically start this launch configuration</td>
<td>Causes the configuration to start automatically each time you start Progress Developer Studio for OpenEdge.</td>
<td></td>
</tr>
<tr>
<td>Automatically restart this launch configuration</td>
<td>Causes the configuration to restart automatically when a procedure running under it terminates. This option is useful if you need to re-initialize static variables. The configuration restarts automatically only if the previously running instance was started in Run (not Debug) mode, and only if Start new AVM is selected in the ABL Virtual Machine section on this tab.</td>
<td></td>
</tr>
<tr>
<td>Copy project settings</td>
<td>Opens a browser that lets you select a project from which to import the settings for project name, TTY, and working directory.</td>
<td></td>
</tr>
<tr>
<td>Apply</td>
<td>Saves the current launch configuration definition.</td>
<td></td>
</tr>
<tr>
<td>Revert</td>
<td>Discards unsaved changes to the current launch configuration definition.</td>
<td></td>
</tr>
</tbody>
</table>
See also
Launch configurations for running and debugging programs on page 551
Using launch configurations on page 562

Startup tab (Application)

The Startup tab in the Progress OpenEdge Application section of the Configurations wizard contains many of the most commonly used startup parameters for an ABL application launch configuration. See Progress OpenEdge Application settings on page 589 for information about accessing the wizard.

The following controls are available on the Startup tab:

<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
<th>Related parameter (if applicable)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong> (accessible from all tabs)</td>
<td>(Required) Specifies the name of the launch configuration, which appears on menus and in the left pane of the Configurations wizard.</td>
<td>Parameter (-p)</td>
</tr>
<tr>
<td><strong>Session startup parameter</strong></td>
<td>Optionally specifies a string to be passed to the AVM as an argument to the -param startup parameter when the configuration starts.</td>
<td>Initialization File (-ininame)</td>
</tr>
<tr>
<td><strong>Ini file</strong></td>
<td>Optionally specifies an initialization file (full path or path relative to the working directory) containing environment settings for the AVM session.</td>
<td></td>
</tr>
<tr>
<td><strong>Use project settings</strong></td>
<td>If checked, causes the configuration to use the same settings as the project (specified on the Main tab) for the assemblies directory and for loading of the Common Language Runtime.</td>
<td></td>
</tr>
<tr>
<td><strong>Assemblies directory</strong></td>
<td>Optionally specifies the directory (full path or path relative to the working directory) containing the assemblies.xml file and any third-party .NET assemblies. If this field is blank, the working directory for the specified project is assumed. This field is disabled if Use project settings is checked.</td>
<td>Assemblies (-assemblies)</td>
</tr>
<tr>
<td><strong>Preload CLR</strong></td>
<td>If checked, causes OpenEdge to load the .NET Common Language Runtime on startup of the configuration.</td>
<td>Preload CLR (-preloadCLR)</td>
</tr>
</tbody>
</table>
### Control | Purpose | Related parameter (if applicable)
--- | --- | ---
Enable Progress Dynamics | If checked, enables use of the Dynamics framework under the configuration. Also:  
- Specify `icfstart.p` as the startup program on the **Main** tab.  
- Include the `-icfparam` parameter in the **Additional startup parameters** field below. | Dynamics Parameter `(-icfparam)`
Configuration file | (Required if Enable Progress Dynamics is selected) Specifies the full or relative (to the working directory) path to the Dynamics `icfconfig.xml` file. |  
Session type | Specifies the Dynamics session type to be used under the configuration. |  
Additional startup parameters | Optionally specifies any desired parameters (with values) not included elsewhere in this configuration definition. Do not include the `-p` parameter in this field. |  
Apply | Saves the current launch configuration definition. |  
Revert | Revert |  
Run or Debug | Runs the specified startup program or opens it in the Progress Developer Studio for OpenEdge Debugger. |  
Close | Closes the **Configurations** wizard. |  

#### See also
- Options for controlling program execution on page 551  
- Options for monitoring results on page 562  
- Options for modifying code while debugging on page 589

#### PROPATH tab (Application)

The PROPATH tab in the Progress OpenEdge Application section of the **Configurations** wizard lets you control where OpenEdge searches for files and procedures when the launch configuration runs. See Progress OpenEdge Application settings on page 589 for information about accessing the wizard.

The PROPATH tab functions like the PROPATH tab in the Project Properties dialog, with two additional command buttons (Add Project and Copy ini Settings) available:
<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong> (accessible from all tabs)</td>
<td>(Required) Specifies the name of the launch configuration, which appears on menus and in the left pane of the <strong>Configurations</strong> wizard.</td>
</tr>
<tr>
<td><strong>Tree view</strong></td>
<td>Shows the current PROPATH definition.</td>
</tr>
<tr>
<td><strong>Add Project</strong></td>
<td>Opens a browser that lets you select a project from which to import PROPATH settings. OpenEdge copies the PROPATH of the project you choose to the selected location in the launch configuration PROPATH.</td>
</tr>
<tr>
<td><strong>Copy ini Settings</strong></td>
<td>Opens a browser that lets you select an INI file from which to import PROPATH settings. OpenEdge replaces the entire existing PROPATH with the PROPATH defined in the selected initialization file.</td>
</tr>
<tr>
<td><strong>Other command buttons</strong></td>
<td>Function as described in the PROPATH help for the <strong>Project Properties</strong> dialog.</td>
</tr>
<tr>
<td><strong>Apply</strong></td>
<td>Saves the current launch configuration definition.</td>
</tr>
<tr>
<td><strong>Revert</strong></td>
<td>Saves the current launch configuration definition.</td>
</tr>
<tr>
<td><strong>Run or Debug</strong></td>
<td>Runs the specified startup program or opens it in the Progress Developer Studio for OpenEdge Debugger.</td>
</tr>
<tr>
<td><strong>Close</strong></td>
<td>Closes the <strong>Configurations</strong> wizard.</td>
</tr>
</tbody>
</table>

**Note:** This tab does not support the use of the OpenEdge `@{ROOT}` substitution variable. However, OpenEdge variables such as `@WORK` and `@DLC` are supported. In addition, the Eclipse substitution variables ([Window > Preferences > Run/Debug > String Substitution](#)) are supported.

### See also

- Launch configurations for running and debugging programs on page 551
- AppServer and WebSpeed debugging on page 558
- Using launch configurations on page 562
- Progress OpenEdge Application settings on page 589
- Progress OpenEdge AppServer and WebSpeed settings on page 604
- Progress External OpenEdge AVM settings Debug Configuration only on page 615

### Databases tab (Application)

The **Databases** tab in the Progress OpenEdge Application section of the **Configurations** wizard lets you specify the OpenEdge database connections used by an ABL application launch configuration. See Progress OpenEdge Application settings on page 589 for information about accessing the wizard.

The **Databases** tab functions like the in the **Project Properties** dialog, with two additional controls (Use project database connections and Show from project) available:
<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name (accessible from all tabs)</strong></td>
<td>(Required) Specifies the name of the launch configuration, which appears on menus and in the left pane of the Configurations wizard.</td>
</tr>
<tr>
<td><strong>Use project database connections</strong></td>
<td>Causes the configuration to use those connections that are used by the project specified on the Main tab for the configuration. Choosing this option disables filtering and selecting from the list.</td>
</tr>
<tr>
<td><strong>Show selected/Show all/Show from project</strong></td>
<td>Filters the database connection list to show: only those connections that are currently used by the configuration; all available connections; or only those connections that are used by the project specified on the Main tab for the configuration.</td>
</tr>
<tr>
<td><strong>Database connection list</strong></td>
<td>Lists available OpenEdge database connections, filtered according to the Show selected/Show all/Show from project selection. The launch configuration uses those entries that are checked.</td>
</tr>
<tr>
<td><strong>Other controls</strong></td>
<td>Function as described in the for the Project Properties dialog.</td>
</tr>
<tr>
<td><strong>Apply</strong></td>
<td>Saves the current launch configuration definition.</td>
</tr>
<tr>
<td><strong>Revert</strong></td>
<td>Revert</td>
</tr>
<tr>
<td><strong>Run or Debug</strong></td>
<td>Runs the specified startup program or opens it in the Progress Developer Studio for OpenEdge Debugger.</td>
</tr>
<tr>
<td><strong>Close</strong></td>
<td>Closes the Configurations wizard.</td>
</tr>
</tbody>
</table>

**See also**
- Launch configurations for running and debugging programs on page 551
- Using launch configurations on page 562
- Progress OpenEdge Application settings on page 589
- Progress OpenEdge AppServer and WebSpeed settings on page 604
- Progress External OpenEdge AVM settings Debug Configuration only on page 615

**ABL tab (Application)**

The ABL tab in the Progress OpenEdge Application section of the Configurations wizard lets you set various options affecting ABL behavior of a launch configuration. See Progress OpenEdge Application settings on page 589 for information about accessing the wizard.

The following controls are available on the ABL tab:
<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
<th>Related parameter (if applicable)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong> (accessible from all tabs)</td>
<td>(Required) Specifies the name of the launch configuration, which appears on menus and in the left pane of the <strong>Configurations</strong> wizard.</td>
<td></td>
</tr>
<tr>
<td><strong>Use system alert message boxes</strong></td>
<td>If checked, causes system messages to be displayed in alert boxes, rather than in the message area, when the configuration runs.</td>
<td></td>
</tr>
<tr>
<td><strong>Use application alert message boxes</strong></td>
<td>If checked, causes application messages to be displayed in alert boxes, rather than in the message area, when the configuration runs.</td>
<td></td>
</tr>
<tr>
<td><strong>Enable debug alert message boxes</strong></td>
<td>If checked, allows users access (via a <strong>Help</strong> button in an alert box and/or a client log) to ABL stack trace information when programs running under the configuration encounter errors.</td>
<td><strong>Debug Alert</strong> (-debugalert)</td>
</tr>
<tr>
<td><strong>Use error stack</strong></td>
<td>If checked, allows error objects to save the ABL call stack in the <strong>CallStack</strong> property.</td>
<td><strong>Error Stack</strong> (-errorstack)</td>
</tr>
<tr>
<td><strong>Reread nolock</strong></td>
<td>If checked, causes the AVM to use the most recent cached copy, when multiple copies exist of the same database record (fetched with the <strong>NO-LOCK</strong> option.)</td>
<td><strong>Reread Nolock</strong> (-rereadnolock)</td>
</tr>
<tr>
<td><strong>Enable use of widget IDs</strong></td>
<td>If checked, allows use of application-defined IDs for ABL widgets.</td>
<td><strong>Use Widget ID</strong> (-usewidgetid)</td>
</tr>
<tr>
<td><strong>Expand browse columns</strong></td>
<td>If checked, causes the AVM to turn on the <strong>EXPANDABLE</strong> and <strong>FIT-LAST-COLUMN</strong> options for browses created in the session.</td>
<td><strong>Expand Browse</strong> (-expandbrow)</td>
</tr>
<tr>
<td><strong>Filter OCX events</strong></td>
<td>If checked, causes the AVM session for the configuration to limit the conditions under which asynchronous COM events are handled.</td>
<td><strong>Filtering Asynchronous COM Events</strong> (-filterocxevents)</td>
</tr>
<tr>
<td><strong>Number of databases</strong></td>
<td>Optionally specifies the maximum number of databases that can be connected during the AVM session.</td>
<td><strong>Number of Databases</strong> (-n)</td>
</tr>
<tr>
<td><strong>Nested blocks</strong></td>
<td>Optionally specifies the maximum number of nested blocks that an application running under the configuration can contain without raising a warning</td>
<td><strong>Nested Blocks</strong> (-nb)</td>
</tr>
<tr>
<td><strong>Input characters</strong></td>
<td>Specifies the maximum number of characters allowed in a single ABL statement.</td>
<td><strong>Input Characters</strong> (-inp)</td>
</tr>
<tr>
<td><strong>Statement tokens</strong></td>
<td>Specifies the maximum number of tokens allowed in a single ABL statement.</td>
<td><strong>Token</strong> (-tok)</td>
</tr>
<tr>
<td><strong>Allow special characters in names</strong></td>
<td>If checked, allows ABL code to use variables or fields whose names contain special characters such as underscores. Enabling this option is required for Dynamics.</td>
<td></td>
</tr>
</tbody>
</table>
### Related parameter (if applicable)

**Use dictionary expressions**

If checked, causes ABL to use dictionary and help validation for all fields in all frames when you compile an application.

Dictionary Expressions (-dictexps)

**Keyword forget file**

Optionally specifies a file (full path or path relative to the working directory) containing a list of ABL keywords that the AVM is to treat as non-keywords.

Keyword Forget List (-k)

**Apply**

Saves the current launch configuration definition.

**Revert**

Discards unsaved changes to the current launch configuration definition.

**Run or Debug**

Runs the specified startup program or opens it in the Progress Developer Studio for OpenEdge Debugger.

**Close**

Closes the **Configurations** wizard.

### See also

- Launch configurations for running and debugging programs on page 551
- Using Launch Configurations on page 562
- Progress OpenEdge Application settings on page 589
- Progress OpenEdge AppServer and WebServer settings on page 604
- OProgress External OpenEdge AVM settings Debug Configuration only on page 615

### Culture tab (Application)

The **Culture** tab in the Progress OpenEdge Application section of the **Configurations** wizard lets you set various internationalization options affecting ABL behavior of a **launch configuration**. See Progress OpenEdge Application settings on page 589 for information about accessing the wizard.

The following controls are available on the **Culture** tab:

<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
<th>Related parameter (if applicable)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong> (accessible from all tabs)</td>
<td>(Required) Specifies the name of the launch configuration, which appears on menus and in the left pane of the <strong>Configurations</strong> wizard.</td>
<td></td>
</tr>
<tr>
<td><strong>Use defaults</strong></td>
<td>If checked, disables all other fields on this tab and causes the AVM to use the values in the startup parameter file(s) used by the configuration.</td>
<td></td>
</tr>
<tr>
<td><strong>Use OS locale</strong></td>
<td>If checked, causes the configuration to use Windows locale settings for numeric and date formats. Any conflicting settings elsewhere on this tab override this setting.</td>
<td>Use OS Locale (-useOsLocale)</td>
</tr>
<tr>
<td><strong>Language</strong></td>
<td>Optionally specifies the initial return value for the <strong>CURRENT-LANGUAGE</strong> function.</td>
<td>Language (-lng)</td>
</tr>
<tr>
<td>Control</td>
<td>Purpose</td>
<td>Related parameter (if applicable)</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------------------------------------------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>Case table</td>
<td>Specifies the case table used by the configuration. Available selections come from the file identified in the Conversion map file field or from the default convmap.cp file.</td>
<td></td>
</tr>
<tr>
<td>Collation table</td>
<td>Specifies the collation table used by the configuration. Available selections come from the file identified in the Conversion map file field or from the default convmap.cp file.</td>
<td></td>
</tr>
<tr>
<td>Date format</td>
<td>Specifies the date-display format used by the configuration.</td>
<td>Date Format (-d)</td>
</tr>
<tr>
<td>Year offset</td>
<td>Specifies the first year of the 100-year period within which two-digit year values are assumed to fall.</td>
<td>Century Year Offset (-yy)</td>
</tr>
<tr>
<td>Fractional separator</td>
<td>Identifies the numeric value (as specified in the selected Internal code page) for the character used as a decimal point in formatted text.</td>
<td>Fractional Separator (-numdec)</td>
</tr>
<tr>
<td>Thousands separator</td>
<td>Identifies the numeric value (as specified in the selected Internal code page) for the character used as a thousands separator in formatted text.</td>
<td>Thousands Separator (-numsep)</td>
</tr>
<tr>
<td>Use 4-digit year</td>
<td>If checked, causes the EXPORT, MESSAGE and PUT UNFORMATTED statements use four-digit values for all years.</td>
<td>Four Digit Year Default (-yr4def)</td>
</tr>
<tr>
<td>Internal code page</td>
<td>Specifies the Internal code page used by the configuration. Available selections come from the file identified in the Conversion map file field or from the default convmap.cp file.</td>
<td>Internal Code Page (-cpinternal)</td>
</tr>
<tr>
<td>Log file code page</td>
<td>Specifies the Log File code page used by the configuration. Available selections come from the file identified in the Conversion map file field or from the default convmap.cp file.</td>
<td>Log File Code Page (-cplog)</td>
</tr>
<tr>
<td>R-code in code page</td>
<td>Specifies the R-code Ocode page used by the configuration. Available selections come from the file identified in the Conversion map file field or from the default convmap.cp file.</td>
<td>R-code In Code Page (-crcodein)</td>
</tr>
<tr>
<td>Terminal code page</td>
<td>Specifies the Terminal code page used by the configuration. Available selections come from the file identified in the Conversion map file field or from the default convmap.cp file.</td>
<td>Terminal Code Page (-cpterms)</td>
</tr>
<tr>
<td>Control</td>
<td>Purpose</td>
<td>Related parameter (if applicable)</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------------------------------------------------------------</td>
<td>--------------------------------------------------------</td>
</tr>
<tr>
<td>Stream code page</td>
<td>Specifies the Stream code page used by the configuration. Available selections come from the file identified in the Conversion map file field or from the default convmap.cp file.</td>
<td>Stream Code Page (-cpstream)</td>
</tr>
<tr>
<td>Print code page</td>
<td>Specifies the Print code page used by the configuration. Available selections come from the file identified in the Conversion map file field or from the default convmap.cp file.</td>
<td>Print Code Page (-cpprint)</td>
</tr>
<tr>
<td>R-code out code page</td>
<td>Specifies the R-code Out code page used by the configuration. Available selections come from the file identified in the Conversion map file field or from the default convmap.cp file.</td>
<td>R-code Out Code Page (-crcodeout)</td>
</tr>
<tr>
<td>Conversion map file</td>
<td>Optionally specifies the conversion map file (full path or path relative to the working directory) used by the configuration. The default convmap.cp file is used if this field is blank.</td>
<td></td>
</tr>
</tbody>
</table>

**See also**

- Launch configurations for running and debugging programs on page 551
- Using launch configurations on page 562
- Progress OpenEdge Application settings on page 589
- Progress OpenEdge AppServer and WebSpeed settings on page 604
- Progress External OpenEdge AVM settings Debug Configuration only on page 615

**Performance tab (Application)**

The Performance tab in the Progress OpenEdge Application section of the Configurations wizard lets you set various options affecting performance of applications running under a launch configuration. See Progress OpenEdge Application settings on page 589 for information about accessing the wizard.

The following controls are available on the Performance tab:
<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
<th>Related parameter (if applicable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name (accessible from all tabs)</td>
<td>(Required) Specifies the name of the launch configuration, which appears on menus and in the left pane of the Configurations wizard.</td>
<td></td>
</tr>
<tr>
<td>Use defaults</td>
<td>If checked, disables all other fields on this tab and causes the AVM to use default values or values specified elsewhere in the configuration definition.</td>
<td></td>
</tr>
<tr>
<td>Temporary files directory</td>
<td>Optionally specifies the directory (full path or path relative to the working directory) for storage of temporary files. The default is the working directory.</td>
<td></td>
</tr>
<tr>
<td>Enforce startup parameter limits</td>
<td>If checked, causes the specified values for directory size, local buffer size, maximum memory, and nested blocks to be enforced as hard limits.</td>
<td>Hardlimit (-hardlimit)</td>
</tr>
<tr>
<td>Schema field cache size</td>
<td>Optionally specifies the number of entries in the schema field cache.</td>
<td>Schema Field Cache Size (-fc)</td>
</tr>
<tr>
<td>Record buffer size</td>
<td>Optionally specifies the size, in bytes, of the standard buffer for database records.</td>
<td>Record Buffer Size (-Mr)</td>
</tr>
<tr>
<td>Stack size</td>
<td>Optionally specifies the size, in kilobytes, of the internal memory area used by ABL program modules.</td>
<td>Stack Size (-s)</td>
</tr>
<tr>
<td>Local buffer size</td>
<td>Optionally specifies the size, in kilobytes, of the local buffer in which the AVM stores all variables, work files, and records that are in use at one time for a user.</td>
<td>Local Buffer Size (-l)</td>
</tr>
<tr>
<td>Directory size</td>
<td>Optionally specifies the number of compiled procedure directory entries.</td>
<td>Directory Size (-D)</td>
</tr>
<tr>
<td>Stash blocks</td>
<td>Optionally specifies the number of 1KB blocks to allocate to the stash area, which OpenEdge uses as temporary storage for modified index fields.</td>
<td>Stash Area (-stsh)</td>
</tr>
<tr>
<td>Temp-table buffer pool size</td>
<td>Optionally specifies the number of buffers in the temporary table database pool.</td>
<td>Number of Buffers for Temporary Tables (-t)</td>
</tr>
<tr>
<td>Speed sort block size</td>
<td>Optionally specifies the block size, in kilobytes, to allocate when sorting records for reports and when rebuilding indexes.</td>
<td>Speed Sort (-TB)</td>
</tr>
<tr>
<td>Speed sort merge streams</td>
<td>Optionally specifies the number of blocks or streams to be simultaneously merged during the sort process.</td>
<td>Merge Number (-TM)</td>
</tr>
<tr>
<td>Async queue size</td>
<td>Optionally specifies the total number of bytes allowed for all buffers allocated when asynchronous requests are queued on behalf of a client.</td>
<td>Async Queue Size (-asyncqueue-size)</td>
</tr>
<tr>
<td>Maximum memory</td>
<td>Optionally specifies the initial amount of memory, in kilobytes, allocated for r-code segments.</td>
<td>Maximum Memory (-mmax)</td>
</tr>
</tbody>
</table>
### Launch and Debugging ABL Programs

<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
<th>Related parameter (if applicable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROLIB Memory</td>
<td>If checked, causes the AVM for the configuration to allocate a 512-byte cache for a standard library directory.</td>
<td>PROLIB Memory (-p1s)</td>
</tr>
<tr>
<td>PROLIB swap</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apply</td>
<td>Saves the current launch configuration definition.</td>
<td></td>
</tr>
<tr>
<td>Revert</td>
<td>Revert</td>
<td></td>
</tr>
<tr>
<td>Run or Debug</td>
<td>Runs the specified startup program or opens it in the Progress Developer Studio for OpenEdge Debugger.</td>
<td></td>
</tr>
<tr>
<td>Close</td>
<td>Closes the Configurations wizard.</td>
<td></td>
</tr>
</tbody>
</table>

**See also**
- Launch configurations for running and debugging programs on page 551
- Using launch configurations on page 562
- Progress OpenEdge Application settings on page 589
- Progress OpenEdge AppServer and WebSpeed settings on page 604
- Progress External OpenEdge AVM settings Debug Configuration only on page 615

**Logging tab (Application)**

The Logging tab in the Progress OpenEdge Application section of the Configurations wizard lets you set various options affecting logging of ABL applications running under a launch configuration. See Progress OpenEdge Application settings on page 589 for information about accessing the wizard.

The following controls are available on the Logging tab:

<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
<th>Related parameter (if applicable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name (accessible from all tabs)</td>
<td>(Required) Specifies the name of the launch configuration, which appears on menus and in the left pane of the Configurations wizard.</td>
<td></td>
</tr>
<tr>
<td>Enable logging</td>
<td>If checked, makes other fields on this tab editable, allowing you to set client logging options.</td>
<td></td>
</tr>
<tr>
<td>Logging level</td>
<td>Specifies the level at which entries are written to the client log file.</td>
<td>Logging Level (-logginglevel)</td>
</tr>
<tr>
<td>Logging file count</td>
<td>Specifies the number of log files, including the current one, to be kept on disk. A value of 0 means no limit on the number of log files.</td>
<td>Number of Log Files to Keep (-numlogfiles)</td>
</tr>
<tr>
<td>Logging size threshold</td>
<td>Specifies the maximum size, in bytes, for log files. When this threshold is reached, the AVM creates a new log file</td>
<td>Log Threshold (-logthreshold)</td>
</tr>
</tbody>
</table>
### Control

<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
<th>Related parameter (if applicable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Append to log file</td>
<td>If checked, causes OpenEdge to add logging information to the most recent existing log file. Deselect this option to create a new log file for each client session.</td>
<td>Record Buffer Size (-Mr)</td>
</tr>
<tr>
<td>Clear log files</td>
<td>If checked, causes OpenEdge, when the session starts, to delete existing log files matching the specified client log file name.</td>
<td>Clear Log (-clearlog)</td>
</tr>
<tr>
<td>Log file</td>
<td>Specifies a name and location (full path or path relative to the working directory) for the client log file.</td>
<td>Client Logging (-clientlog)</td>
</tr>
<tr>
<td>Log entry list</td>
<td>Specifies the types of events to be logged. In the Log Entry Type column, check the items to be included. For each such entry type, click in the Log Entry Level column and select the logging level.</td>
<td></td>
</tr>
<tr>
<td>Apply</td>
<td>Saves the current launch configuration definition.</td>
<td></td>
</tr>
<tr>
<td>Revert</td>
<td>Revert</td>
<td></td>
</tr>
<tr>
<td>Run or Debug</td>
<td>Runs the specified startup program or opens it in the Progress Developer Studio for OpenEdge Debugger.</td>
<td></td>
</tr>
<tr>
<td>Close</td>
<td>Closes the Configurations wizard.</td>
<td></td>
</tr>
</tbody>
</table>

### See also

- Launch configurations for running and debugging programs on page 551
- Using launch configurations on page 562
- Progress OpenEdge Application settings on page 589
- Progress OpenEdge AppServer and WebSpeed settings on page 604
- Progress External OpenEdge AVM settings Debug Configuration only on page 615

### Environment tab (Application)

The **Environment** tab in the Progress OpenEdge Application section of the Configurations wizard lets you define the environment variables used by a launch configuration. You can use the variables defined on this tab either in addition to or in place of those defined for the operating system. See Progress OpenEdge Application settings on page 589 for information about accessing the wizard.

The following controls are available on the **Environment** tab:

<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name (accessible from all tabs)</td>
<td>(Required) Specifies the name of the launch configuration, which appears on menus and in the left pane of the Configurations wizard.</td>
</tr>
<tr>
<td>Variables grid</td>
<td>Lists the names and values of environment variables currently defined for the configuration.</td>
</tr>
</tbody>
</table>
PurposeControl

New
Opens a dialog that lets you define a name/value pair as a new environment variable, which is added to the current list in the grid.

Select
Opens a browser that lets you copy selected environment variables in use by the operating system. Check the desired entries and click OK to add them to the list in the grid.

Edit
Opens a dialog that lets you modify the name/value of the environment variable currently selected in the grid.

Remove
Deletes the currently selected environment variable from the grid.

Append environment to native environment/Replace native environment with specified environment
Tells the configuration to use the environment variables specified on this tab either in addition to or in place of those defined for the operating system.

Apply
Saves the current launch configuration definition.

Revert
Revert

Run or Debug
Runs the specified startup program or opens it in the Progress Developer Studio for OpenEdge Debugger.

Close
Closes the Configurations wizard.

See also
Launch configurations for running and debugging programs on page 551
Using launch configurations on page 562
Progress OpenEdge Application settings on page 589
Progress OpenEdge AppServer and WebSpeed settings on page 604
Progress External OpenEdge AVM settings Debug Configuration only on page 615

Common tab (Application)
The Common tab in the Progress OpenEdge Application section of the Configurations wizard allows you to set options for specifying where an ABL application launch configuration is saved, how it is accessed, and how it runs. See Progress OpenEdge Application settings on page 589 for information about accessing the wizard.

The following controls are available on the Common tab:

<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name (accessible from all tabs)</td>
<td>(Required) Specifies the name of the launch configuration, which appears on menus and in the left pane of the Configurations wizard.</td>
</tr>
<tr>
<td>Local file/Shared file</td>
<td>Notifies Progress Developer Studio for OpenEdge to save the configuration definition either in the metadata\plugins directory in your workspace (local file), or in a selected project folder (shared file).</td>
</tr>
<tr>
<td>Control</td>
<td>Purpose</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Favorites menu selector</td>
<td>Causes the configuration to appear as an option at the top level of the Run menu, the Debug menu, or both. Check the desired options.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> You can also manage these menu options on via Organize Favorites on the Run and Debug menus.</td>
</tr>
<tr>
<td>Encoding: Default/Other</td>
<td>Specifies the code page used for program output to the Eclipse console, either the default or a selected non-default code page.</td>
</tr>
<tr>
<td>Allocate Console</td>
<td>If checked, allows program output to appear in the Eclipse console.</td>
</tr>
<tr>
<td>File</td>
<td>If checked, redirects program output to the specified file.</td>
</tr>
<tr>
<td>Append</td>
<td>If checked, notifies Progress Developer Studio for OpenEdge to append redirected program output to the specified file. Otherwise, the file is overwritten.</td>
</tr>
<tr>
<td>Launch in background</td>
<td>If checked, lets you perform other tasks while the configuration runs.</td>
</tr>
<tr>
<td>Apply</td>
<td>Saves the current launch configuration definition.</td>
</tr>
<tr>
<td>Revert</td>
<td>Revert</td>
</tr>
<tr>
<td>Run or Debug</td>
<td>Runs the specified startup program or opens it in the Progress Developer Studio for OpenEdge Debugger.</td>
</tr>
<tr>
<td>Close</td>
<td>Closes the Configurations wizard.</td>
</tr>
</tbody>
</table>

### See also
- Launch configurations for running and debugging programs on page 551
- Using launch configurations on page 562
- Progress OpenEdge Application settings on page 589
- Progress OpenEdge AppServer and WebSpeed settings on page 604
- Progress External OpenEdge AVM settings Debug Configuration only on page 615

### Profiler tab (Application)

The **Profiler** tab in the Progress OpenEdge Application section of the Configurations wizard contains the options to run the profiler for any session. See Progress OpenEdge Application settings on page 589 for information about accessing the wizard.

The following controls are available on the **Profiler** tab:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
<th>Related parameter (if applicable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable profiling</td>
<td>Enables profiling for the selected session</td>
<td>-profile</td>
</tr>
<tr>
<td>Profiler output folder</td>
<td>Specifies a common location for the output folder for a profiled output file and a listing file.</td>
<td>-OUTFILE for the location for profiler output file and -LISP the files directory location</td>
</tr>
</tbody>
</table>
### Related parameter (if applicable)

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
<th>Related parameter (if applicable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Specifies a description for the session profiling.</td>
<td>-description</td>
</tr>
<tr>
<td>Generate debug listing files</td>
<td>Generates the debug listing files in the profiler output directory. This file provides a detailed line information in the Line summary view. If you do not add this option, no information is displayed in the Line summary view.</td>
<td>-LISTING</td>
</tr>
<tr>
<td>Enable tracing</td>
<td>Generates a detailed tracing data that is used to generate call tree in the profiler viewer. If you do not add this option, no information is displayed in the Call Tree tab.</td>
<td>-TRACE-FILTER</td>
</tr>
</tbody>
</table>

**Note:** It generates a very large output file.

**Note:** When you click Apply, the session is profiled and the .prof file opens in the editor.

### Progress OpenEdge AppServer and WebSpeed settings

The topics in this section describe the options and controls available for defining Progress OpenEdge AppServer and Progress OpenEdge WebSpeed launch configurations in the Configurations wizard.

This wizard appears when you select Run Configurations or Debug Configurations from:

- The drop-down menu on either the Run button or the Debug button
- The ABL Editor context (right-click) menu

Although the window title includes either "Run" or "Debug," you have access to all launch configurations. The options are identical regardless of how you open the wizard.

Various options have the effect of invoking a startup parameter for the AVM session. In such cases, the relevant parameter is shown in the rightmost column of the table. For more information, see OpenEdge Deployment: Startup Command and Parameter Reference, available at the Progress Product Documentation Web site.

**Note:** The launch configuration settings are common to both Progress OpenEdge AppServer and Progress OpenEdge WebSpeed in all the tabs except the Startup tab. For more information, see Startup tab on page 605.

There is a separate topic for each of the tabs in this section of the wizard.

### Server tab

The Server tab in the Progress OpenEdge AppServer, Progress OpenEdge WebSpeed, and Progress Application Server for OpenEdge sections of the Configurations wizard specifies the AppServer or WebSpeed instance used by the launch configuration. See Progress OpenEdge AppServer and WebSpeed settings on page 604 for information about accessing the wizard.

The following controls are available on the Server tab:
### Purpose

<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong> (accessible from all tabs)</td>
<td>(Required) Specifies the name of the launch configuration, which appears on menus and in the left pane of the <strong>Configurations</strong> wizard.</td>
</tr>
<tr>
<td><strong>Server</strong></td>
<td>(Required) Specifies the AppServer or WebSpeed instance that starts when the launch configuration runs.</td>
</tr>
<tr>
<td><strong>Runtime Environment</strong></td>
<td>(Read-only) Specifies the OpenEdge version of the AppServer or WebSpeed instance. You can change this value if multiple OpenEdge AVM runtimes are installed on the local machine.</td>
</tr>
<tr>
<td><strong>Host Name</strong></td>
<td>(Read-only) Specifies the machine on which the AppServer or WebSpeed broker runs. Progress Developer Studio for OpenEdge currently provides AppServer and WebSpeed support on both local and remote machines.</td>
</tr>
<tr>
<td><strong>Apply</strong></td>
<td>Saves the current launch configuration definition.</td>
</tr>
<tr>
<td><strong>Revert</strong></td>
<td>Discards unsaved changes to the current launch configuration definition.</td>
</tr>
<tr>
<td><strong>Run or Debug</strong></td>
<td>Starts or debugs the specified instance.</td>
</tr>
<tr>
<td><strong>Close</strong></td>
<td>Closes the <strong>Configurations</strong> wizard.</td>
</tr>
</tbody>
</table>

### See also
- Launch configurations for running and debugging programs on page 551
- Running and debugging ABL programs on page 551
- Managing launch configurations on page 566
- Starting an AppServer or a WebSpeed instance on page 564
- Progress OpenEdge AppServer and WebSpeed settings on page 604
- Progress OpenEdge Application settings on page 589
- Progress External OpenEdge AVM settings Debug Configuration only on page 615

### Startup tab

The **Startup** tab in the Progress OpenEdge AppServer, Progress OpenEdge WebSpeed, Progress Application Server for OpenEdge, and ABLUnit sections of the **Configurations** wizard specifies options for starting the AppServer and WebSpeed instances used by the launch configuration.

The following controls are available on the **Startup** tab:

<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong> (accessible from all tabs)</td>
<td>(Required) Specifies the name of the launch configuration, which appears on menus and in the left pane of the <strong>Configurations</strong> wizard.</td>
</tr>
<tr>
<td>Control</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Working directory</td>
<td>Specifies the directory in which the AVM starts when the configuration runs. By default, the value is set to the OpenEdge installation working directory. If you created the launch configuration by using the New Server wizard, the value you set for the AppServer or WebSpeed broker is used. Click Workspace, File System, or Variables to specify a location by browsing or using a variable.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong></td>
</tr>
<tr>
<td></td>
<td>For remote AppServer or WebSpeed launch configurations, wherever a path is expected, the File System or the Workspace options are disabled. There is no support for remote host directory browsing. When you select Apply or OK, the Debugger validates the specified properties but does not update them to the broker until the broker is started.</td>
</tr>
<tr>
<td>Activate (Progress OpenEdge AppServer only)</td>
<td>(Optional) Specifies the procedure that is to run when each of these AppServer operations occurs. Click Browse to display and select from the Select Activation Procedure dialog. This dialog includes an option to filter the list to include only those ABL procedures that are both in folders designated as AppServer modules and in projects associated with the server.</td>
</tr>
<tr>
<td>Deactivate (Progress OpenEdge AppServer only)</td>
<td>(Optional) Specifies one or more startup parameters to be used when the startup procedure, if any, runs.</td>
</tr>
<tr>
<td>Connect (Progress OpenEdge AppServer only)</td>
<td>(Optional) Specifies one or more startup parameters to be used when the server instance starts.</td>
</tr>
<tr>
<td>Disconnect (Progress OpenEdge AppServer only)</td>
<td></td>
</tr>
<tr>
<td>Startup (Progress OpenEdge AppServer only)</td>
<td></td>
</tr>
<tr>
<td>Shutdown (Progress OpenEdge AppServer only)</td>
<td></td>
</tr>
<tr>
<td>Startup procedure parameters (Progress OpenEdge AppServer only)</td>
<td></td>
</tr>
<tr>
<td>Agent startup parameters</td>
<td></td>
</tr>
</tbody>
</table>
The Debugger tab in the Progress OpenEdge AppServer, Progress OpenEdge WebSpeed, Progress Application Server for OpenEdge, and ABLUnit sections of the Debug Configurations wizard provides options for:

- Specifying connection properties
- Attaching the Debugger to a broker
- Specifying secure client connection properties

The following controls are available on the Debugger tab:

<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong> (accessible from all tabs)</td>
<td>(Required) Specifies the name of the launch configuration, which appears on menus and in the left pane of the Configurations wizard.</td>
</tr>
<tr>
<td>Debug port</td>
<td>Specifies the debug port number on which the debug clients are connected to the broker.</td>
</tr>
<tr>
<td>Password</td>
<td>(Optional) Allows the broker to accept debug connections only from authenticated clients. If specified, the debug clients must use the same password when connecting to the broker for debugging.</td>
</tr>
<tr>
<td>Control</td>
<td>Purpose</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Attach to Developer Studio for debugging</strong></td>
<td><em>(Optional)</em> Attaches the Progress Developer Studio for OpenEdge Debugger to the broker. If this option is selected, you must specify the source lookup paths on the Source tab (See Edit Source Lookup Path dialog on page 624). If it is not selected, other debug clients can connect to the broker for debugging.</td>
</tr>
<tr>
<td><strong>Debugger Client Certificate Path</strong></td>
<td>Specifies the certificate location of the client that is attached to the Debugger. This control is enabled when you select <em>Attach to Developer Studio for debugging</em> and <em>Enable SSL for broker debugger</em>.</td>
</tr>
<tr>
<td><strong>Do not verify host</strong></td>
<td><em>(Optional)</em> Specifies not to verify the host while establishing the connection. <strong>Note:</strong> Host verification happens by default when you enable SSL. This control is enabled when you select <em>Attach to Developer Studio for debugging</em> and <em>Enable SSL for broker debugger</em>.</td>
</tr>
<tr>
<td><strong>Enable SSL for broker debugger</strong></td>
<td><em>(Optional)</em> Specifies that all connections to the broker must use SSL tunneling. An SSL client encrypts data using a private key or a digital certificate, thereby making sure that both the debug client and the broker can authenticate each other's identity. If you are sending data to a remote database, you can ensure that communication between the two machines is secure.</td>
</tr>
<tr>
<td><strong>Broker private key/digital certificate alias name</strong></td>
<td><em>(Optional)</em> Specifies the alias name within the OpenEdge keystore of the private key and digital certificate entry to be used to authenticate all connections to the broker. If this is not selected, the broker uses the <code>default_server</code> server certificate alias. You must provide a unique alias for each certificate or private key.</td>
</tr>
<tr>
<td><strong>Password to access the key/certificate</strong></td>
<td><em>(Optional)</em> Specifies the password to be used for accessing the private key and digital certificate. <strong>Note:</strong> If you use the <code>default_server</code> server certificate, it also has a default password that you need not specify.</td>
</tr>
</tbody>
</table>
**Control** | **Purpose**
---|---
**Import From Server** | Copies the Startup tab values from the AppServer or WebSpeed broker stored with the AdminServer. Note that PROPATH and database information are not copied. It also populates the debug port and password values from the server properties (ubroker.properties) file.

**Apply** | Saves the current launch configuration definition.

**Revert** | Discards unsaved changes to the current launch configuration definition.

**Debug** | Debugs the specified AppServer or WebSpeed instance.

**Close** | Closes the Configurations wizard.

**See also**
- Launch configurations for running and debugging programs on page 551
- Running and debugging ABL programs on page 551
- Managing launch configurations on page 566
- Starting an AppServer or a WebSpeed instance on page 564
- Progress OpenEdge AppServer and WebSpeed settings on page 604
- Progress OpenEdge Application settings on page 589
- Progress External OpenEdge AVM settings Debug Configuration only on page 615

**PROPATH tab**

The PROPATH tab in the Progress OpenEdge AppServer, Progress OpenEdge WebSpeed, Progress Application Server for OpenEdge, and ABLUnit sections of the Configurations wizard allows you to control where OpenEdge searches for files and procedures when the launch configuration runs.

The PROPATH tab functions like the PROPATH tab in the Project Properties dialog, with two additional command buttons (Add Project and Copy ini Settings) available:

<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong> (accessible from all tabs)</td>
<td>(Required) Specifies the name of the launch configuration, which appears on menus and in the left pane of the Configurations wizard.</td>
</tr>
<tr>
<td><strong>Tree view</strong></td>
<td>Shows the current PROPATH definition.</td>
</tr>
<tr>
<td><strong>Add Project</strong></td>
<td>Opens a browser that lets you select a project from which to import PROPATH settings. OpenEdge copies the PROPATH of the project you choose to the selected location in the launch configuration PROPATH.</td>
</tr>
<tr>
<td><strong>Copy ini Settings</strong></td>
<td>Opens a browser that lets you select an INI file from which to import PROPATH settings. OpenEdge replaces the entire existing PROPATH with the PROPATH defined in the selected initialization file.</td>
</tr>
</tbody>
</table>
Databasestab

The Databases tab in the Progress OpenEdge AppServer, Progress OpenEdge WebSpeed, Progress Application Server for OpenEdge, and ABLUnit sections of the Configurations wizard allows you to specify the OpenEdge database connections used by launch configuration.

The Databases tab functions like the in the Project Properties dialog, with the following controls available:

<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name (accessible from all tabs)</td>
<td>(Required) Specifies the name of the launch configuration, which appears on menus and in the left pane of the Configurations wizard.</td>
</tr>
<tr>
<td>Show selected/Show all</td>
<td>Filters the database connection list to show: only those connections that are currently used by the configuration, or all available connections.</td>
</tr>
<tr>
<td>Database connection list</td>
<td>Lists available OpenEdge database connections, filtered according to the Show selected/Show all/Show from project selection. The launch configuration uses those entries that are checked.</td>
</tr>
<tr>
<td>Other controls</td>
<td>Function as described in the for the Project Properties dialog.</td>
</tr>
<tr>
<td>Apply</td>
<td>Saves the current launch configuration definition.</td>
</tr>
<tr>
<td>Control</td>
<td>Purpose</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Revert</td>
<td>Discards unsaved changes to the current launch configuration definition.</td>
</tr>
<tr>
<td>Run or Debug</td>
<td>Starts or debugs the specified instance.</td>
</tr>
<tr>
<td>Close</td>
<td>Closes the Configurations wizard.</td>
</tr>
</tbody>
</table>

**See also**

- [Launch configurations for running and debugging programs](#) on page 551
- [Running and debugging ABL programs](#) on page 551
- [Managing launch configurations](#) on page 566
- [Starting an AppServer or a WebSpeed instance](#) on page 564
- [Progress OpenEdge AppServer and WebSpeed settings](#) on page 604
- [Progress OpenEdge Application settings](#) on page 589
- [Progress External OpenEdge AVM settings Debug Configuration only](#) on page 615

**Security tab**

The **Security** tab in the Progress OpenEdge AppServer, Progress OpenEdge WebSpeed, and Progress Application Server for OpenEdge sections of the Configurations wizard allows you to specify security parameters for the AppServer and WebSpeed instances used by the launch configuration. See [Progress OpenEdge AppServer and WebSpeed settings](#) on page 604 for information about accessing the wizard.

You can use the **Security** tab to secure client connection properties. The following controls are available on the tab:

<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong> (accessible from all tabs)</td>
<td>(Required) Specifies the name of the launch configuration, which appears on menus and in the left pane of the Configurations wizard.</td>
</tr>
<tr>
<td><strong>Enable SSL client connections</strong></td>
<td>(Optional) Specifies that all connections to the broker must use SSL tunneling. An SSL client encrypts data using a private key or a digital certificate, thereby making sure that both the debug client and the broker can authenticate each other’s identity. If you are sending data to a remote database, you can ensure that communication between the two machines is secure.</td>
</tr>
<tr>
<td><strong>Private key/digital certificate alias name</strong></td>
<td>(Optional) The alias name within the OpenEdge keystore of the private key and digital certificate entry to be used to authenticate all connections to the broker. If this is not selected, the broker uses the default_server server certificate alias. You must provide a unique alias for each certificate or private key.</td>
</tr>
</tbody>
</table>
## Control and Purpose

<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Password to access the key/certificate</strong></td>
<td>(Optional) The password to be used for accessing the private key and digital certificate.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>If you use the default_server server certificate, it also has a default password that you need not specify.</td>
</tr>
<tr>
<td><strong>Disable SSL session caching</strong></td>
<td>(Optional) Disables caching for the SSL client session.</td>
</tr>
<tr>
<td><strong>SSL session cache timeout</strong></td>
<td>(Optional) The duration, in seconds, that an SSL client session is held in the session cache, during which an SSL client can resume its session. The default value is 180.</td>
</tr>
<tr>
<td><strong>Apply</strong></td>
<td>Saves the current launch configuration definition.</td>
</tr>
<tr>
<td><strong>Revert</strong></td>
<td>Discards unsaved changes to the current launch configuration definition.</td>
</tr>
<tr>
<td><strong>Run or Debug</strong></td>
<td>Starts or debugs the specified instance.</td>
</tr>
<tr>
<td><strong>Close</strong></td>
<td>Closes the <strong>Configurations</strong> wizard.</td>
</tr>
</tbody>
</table>

### See also
- Launch configurations for running and debugging programs on page 551
- Running and debugging ABL programs on page 551
- Managing launch configurations on page 566
- Starting an AppServer or a WebSpeed instance on page 564
- Progress OpenEdge AppServer and WebSpeed settings on page 604
- Progress OpenEdge Application settings on page 589
- Progress External OpenEdge AVM settings Debug Configuration only on page 615

### Source tab (Progress Source tab)

The **Source** tab in the Progress OpenEdge AppServer and Progress OpenEdge WebSpeed sections of the Configurations wizard allows you to specify the source lookup paths for the AppServer or WebSpeed launch configuration. See Progress OpenEdge AppServer and WebSpeed settings on page 604 for information about accessing the wizard.

The following controls are available on the **Source** tab:

<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong> (accessible from all tabs)</td>
<td>(Required) Specifies the name of the launch configuration, which appears on menus and in the left pane of the <strong>Configurations</strong> wizard.</td>
</tr>
</tbody>
</table>
### Purpose

Specifies one or more source lookup paths where Progress Developer Studio for OpenEdge searches for the source code of the program being executed by the attached AVM.

The default source lookup path includes `$DLC\src` and all the workspace project folders in alphabetical order.

**Note:** Progress Developer Studio for OpenEdge searches for the source file in the source lookup paths in the order in which they are displayed.

### Controls

<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Lookup Path</td>
<td>Specifies one or more source lookup paths where Progress Developer Studio for OpenEdge searches for the source code of the program being executed by the attached AVM. The default source lookup path includes <code>$DLC\src</code> and all the workspace project folders in alphabetical order.</td>
</tr>
<tr>
<td>Add</td>
<td>Launches the Add Source dialog where you can add custom source lookup paths. See <a href="#">Add Source dialog</a> on page 623 for more information.</td>
</tr>
<tr>
<td>Edit</td>
<td>Changes the value of the current entry.</td>
</tr>
<tr>
<td>Remove</td>
<td>Deletes an entry from the list.</td>
</tr>
<tr>
<td>Up, Down</td>
<td>Changes the position in the list of a current entry, and thus changes the search order.</td>
</tr>
<tr>
<td>Restore Default</td>
<td>Resets the source lookup path entries to default.</td>
</tr>
<tr>
<td>Search for duplicate source files on the path</td>
<td>Includes source files with duplicate names while searching in the source lookup paths.</td>
</tr>
<tr>
<td>Apply</td>
<td>Saves the current launch configuration definition.</td>
</tr>
<tr>
<td>Revert</td>
<td>Discards unsaved changes to the current launch configuration definition.</td>
</tr>
<tr>
<td>Run or Debug</td>
<td>Starts or debugs the specified instance.</td>
</tr>
<tr>
<td>Close</td>
<td>Closes the Configurations wizard.</td>
</tr>
</tbody>
</table>

### See also

- Launch configurations for running and debugging programs on page 551
- Running and debugging ABL programs on page 551
- Managing launch configurations on page 566
- Starting an AppServer or a WebSpeed instance on page 564
- Progress OpenEdge AppServer and WebSpeed settings on page 604
- Progress OpenEdge Application settings on page 589
- Progress External OpenEdge AVM settings Debug Configuration only on page 615
Common tab

The Common tab in the Progress OpenEdge AppServer, Progress OpenEdge WebSpeed, Progress Application Server for OpenEdge, and ABLUnit sections of the Configurations wizard allows you to set options for specifying where an AppServer or WebSpeed launch configuration is saved, how it is accessed, and how it runs. See Progress OpenEdge AppServer and WebSpeed settings on page 604 for information about accessing the wizard.

The following controls are available on the Common tab:

<table>
<thead>
<tr>
<th>Control Common tab</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name (accessible from all tabs)</td>
<td>(Required) Specifies the name of the launch configuration, which appears on menus and in the left pane of the Configurations wizard.</td>
</tr>
<tr>
<td>Local file/Shared file</td>
<td>Notifies Progress Developer Studio for OpenEdge to save the configuration definition either in the metadata\plugins directory in your workspace (local file), or in a selected project folder (shared file).</td>
</tr>
<tr>
<td>Favorites menu selector</td>
<td>Causes the configuration to appear as an option at the top level of the Run menu, the Debug menu, or both. Check the desired options.</td>
</tr>
<tr>
<td>Note:</td>
<td>You can also manage these menu options on via Organize Favorites on the Run and Debug menus.</td>
</tr>
<tr>
<td>Encoding: Default/Other</td>
<td>Specifies the code page used for program output to the Eclipse console, either the default or a selected non-default code page.</td>
</tr>
<tr>
<td>Allocate Console</td>
<td>If checked, allows program output to appear in the Eclipse console.</td>
</tr>
<tr>
<td>File</td>
<td>If checked, redirects program output to the specified file.</td>
</tr>
<tr>
<td>Append</td>
<td>If checked, notifies Progress Developer Studio for OpenEdge to append redirected program output to the specified file. Otherwise, the file is overwritten.</td>
</tr>
<tr>
<td>Launch in background</td>
<td>If checked, lets you perform other tasks while the configuration runs.</td>
</tr>
<tr>
<td>Apply</td>
<td>Saves the current launch configuration definition.</td>
</tr>
<tr>
<td>Revert</td>
<td>Discards unsaved changes to the current launch configuration definition.</td>
</tr>
<tr>
<td>Run or Debug</td>
<td>Starts or debugs the specified instance.</td>
</tr>
<tr>
<td>Close</td>
<td>Closes the Configurations wizard.</td>
</tr>
</tbody>
</table>

See also

Launch configurations for running and debugging programs on page 551
Running and debugging ABL programs on page 551
Managing launch configurations on page 566
Starting an AppServer or a WebSpeed instance on page 564
Progress OpenEdge AppServer and WebSpeed settings on page 604
Progress OpenEdge Application settings on page 589
Progress External OpenEdge AVM settings (Debug Configuration only)

The topics in this section describe the options and controls available for defining debug launch configurations for an external OpenEdge AVM. These launch configurations are available in the Progress External OpenEdge AVM section of the Debug Configurations wizard.

To access this wizard, select Debug Configurations from:

- The drop-down menu on the Debug button
- The ABL Editor context (right-click) menu

Various options have the effect of invoking a startup parameter for the AVM session. In such cases, the relevant parameter is shown in the rightmost column of the table. See OpenEdge Deployment: Startup Command and Parameter Reference, available at the Progress Product Documentation Web site, for detailed information.

There is a separate topic for each of the tabs in this section of the wizard.

Connection tab (Progress OpenEdge External AVM)

The Connection tab in the Progress OpenEdge External AVM section of the Debug Configurations wizard contains the required settings for defining an external AVM debug launch configuration, as well as some other frequently used options for debugging an external AVM. See Progress External OpenEdge AVM settings (Debug Configuration only) on page 615 for information on accessing the Debug Configurations wizard.

The following controls are available on the Connection tab:

<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name (accessible from all tabs)</td>
<td>(Required) The name of the launch configuration, which appears on menus and in the left pane of the Configurations wizard.</td>
</tr>
<tr>
<td>Connection type</td>
<td>Specifies the connection type of the application to be debugged.</td>
</tr>
<tr>
<td>Make debug-ready and attach (local host only)</td>
<td>Makes a process debug-ready and attaches the Debugger to the process. Select this option only if the process is running on the same machine as the Debugger.</td>
</tr>
<tr>
<td>Process ID</td>
<td>Specifies the ID of the process on the local machine which you want to debug.</td>
</tr>
<tr>
<td>Select Process</td>
<td>(Optional) Displays the list of currently running OpenEdge processes on the same machine as the Debugger, in the Select Process dialog on page 623.</td>
</tr>
<tr>
<td>Port</td>
<td>(On local machine) Specifies the port number which you want to use for communicating with the client process. The default value (0) instructs the AVM to determine a usable port number.</td>
</tr>
<tr>
<td>Attach to debug-ready process</td>
<td>Attaches the Debugger to a process you have already made debug-ready. Select this option if the AVM instance is running on a remote machine.</td>
</tr>
<tr>
<td>Host</td>
<td>Specifies the name of the host on which the process is running. You can accept the default localhost. You must enter a host name if the process is not running on the same machine as the Debugger.</td>
</tr>
</tbody>
</table>
### Source tab (Progress OpenEdge External AVM)

The **Source** tab in the Progress OpenEdge External AVM section of the **Debug Configurations** wizard contains the required settings for defining an external OpenEdge AVM debug launch configuration, and some other frequently used options for debugging an external AVM. See Progress External OpenEdge AVM settings (Debug Configuration only) on page 615 for information on accessing the **Debug Configurations** wizard.

You can use the **Source** tab to add, edit, and remove source lookup entries.

The following controls are available on the **Source** tab:

<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debug Port</td>
<td>(On remote machine) Specifies the port number which was used to make the process debug-ready.</td>
</tr>
<tr>
<td>Application Name</td>
<td>Specifies the name of the application.</td>
</tr>
<tr>
<td>Note:</td>
<td>This applies only to Progress Application Server for OpenEdge.</td>
</tr>
<tr>
<td>Enable SSL</td>
<td>Specifies that all connections must use SSL tunneling.</td>
</tr>
<tr>
<td>Certificate Path</td>
<td>Specifies the certificate location of the client that is attached to the Debugger.</td>
</tr>
<tr>
<td>Apply</td>
<td>Saves the current launch configuration definition.</td>
</tr>
<tr>
<td>Revert</td>
<td>Discards unsaved changes to the current launch configuration definition.</td>
</tr>
<tr>
<td>Debug</td>
<td>Debugs the specified external OpenEdge AVM.</td>
</tr>
<tr>
<td>Close</td>
<td>Closes the <strong>Configurations</strong> wizard.</td>
</tr>
</tbody>
</table>

### See also
- Attachable Debugger on page 557
- AppServer and WebSpeed debugging on page 558
- Launch configurations for running and debugging programs on page 551
- Running and debugging ABL programs on page 551
- Attaching the Debugger to an external AVM on page 572
- Managing launch configurations on page 566
- Starting an AppServer or a WebSpeed instance on page 564
- Progress External OpenEdge AVM settings Debug Configuration only on page 615
- Progress OpenEdge Application settings on page 589
- Progress OpenEdge AppServer and WebSpeed settings on page 604
- Select Process dialog on page 623
<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name (accessible from all tabs)</td>
<td>(Required) The name of the launch configuration, which appears on menus and in the left pane of the <strong>Configurations</strong> wizard.</td>
</tr>
</tbody>
</table>
| Source Lookup Path                  | Specifies one or more source lookup paths where Progress Developer Studio for OpenEdge searches for the source code of the program being executed by the attached AVM.  
|                                    | The default source lookup path includes `$DLC\src` and all the workspace project folders in alphabetical order.                     |
| **Note:**                          | Progress Developer Studio for OpenEdge searches for the source file in the source lookup paths in the order in which they are displayed. |
| Add                                 | Launches the **Add Source** dialog where you can add custom source lookup paths. See **Add Source dialog** on page 623 for more information. |
| Edit                                | Changes the value of the current entry.                                                                                                                                                        |
| Remove                              | Deletes an entry from the list.                                                                                                                                                                 |
| Up, Down                            | Changes the position in the list of a current entry, and thus changes the search order.                                                                                                          |
| Restore Default                     | Resets the source lookup path entries to default.                                                                                                                                                |
| Search for duplicate source files on the path | Includes source files with duplicate names while searching in the source lookup paths.                                                                                                          |
| Apply                               | Saves the current launch configuration definition.                                                                                                                                               |
| Revert                              | Discards unsaved changes to the current launch configuration definition.                                                                                                                          |
| Debug                               | Debugs the specified external OpenEdge AVM.                                                                                                                                                       |
| Close                               | Closes the **Configurations** wizard.                                                                                                                                                            |

**See also**
- Attachable Debugger on page 557
- AppServer and WebSpeed debugging on page 558
- Launch configurations for running and debugging programs on page 551
- Running and debugging ABL programs on page 551
- Attaching the Debugger to an external AVM on page 572
- Managing launch configurations on page 566
- Starting an AppServer or a WebSpeed instance on page 564
**Common tab (Progress OpenEdge External AVM)**

The **Common** tab in the Progress OpenEdge External AVM section of the **Configurations** wizard allows you to set options for specifying where the external AVM **launch configuration** is saved, how it is accessed, and how it runs. See **Progress External OpenEdge AVM settings (Debug Configuration only)** on page 615 for information about accessing the wizard.

The following controls are available on the **Common** tab:

<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name (accessible from all tabs)</td>
<td>(Required) The name of the launch configuration, which appears on menus and in the left pane of the <strong>Configurations</strong> wizard.</td>
</tr>
<tr>
<td>Local file/Shared file</td>
<td>Notifies Progress Developer Studio for OpenEdge to save the configuration definition either in the <code>metadata\plugins</code> directory in your workspace (local file), or in a selected project folder (shared file).</td>
</tr>
<tr>
<td>Favorites menu selector</td>
<td>Causes the configuration to appear as an option at the top level of the Run menu, the <strong>Debug</strong> menu, or both. Check the desired options.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> You can also manage these menu options on via <strong>Organize Favorites</strong> on the Run and Debug menus.</td>
</tr>
<tr>
<td>Encoding: Default/Other</td>
<td>Specifies the code page used for program output to the Eclipse console, either the default or a selected non-default code page.</td>
</tr>
<tr>
<td>Allocate Console</td>
<td>If checked, allows program output to appear in the Eclipse console.</td>
</tr>
<tr>
<td>File</td>
<td>If checked, redirects program output to the specified file.</td>
</tr>
<tr>
<td>Append</td>
<td>If checked, notifies Progress Developer Studio for OpenEdge to append redirected program output to the specified file. Otherwise, the file is overwritten.</td>
</tr>
<tr>
<td>Launch in background</td>
<td>If checked, lets you perform other tasks while the configuration runs.</td>
</tr>
<tr>
<td>Apply</td>
<td>Saves the current launch configuration definition.</td>
</tr>
<tr>
<td>Revert</td>
<td>Discards unsaved changes to the current launch configuration definition.</td>
</tr>
<tr>
<td>Debug</td>
<td>Debugs the specified external OpenEdge AVM.</td>
</tr>
<tr>
<td>Close</td>
<td>Closes the <strong>Configurations</strong> wizard.</td>
</tr>
</tbody>
</table>

**See also**

Launch configurations for running and debugging programs on page 551
Launching preferences

The Launching page (Progress OpenEdge > Advanced > Launching) of the OpenEdge lets you select the default behavior of new launch configurations with respect to two key options. It includes the following controls:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use project AVM for new launch configurations</td>
<td>If checked, causes new launch configurations to use the instance of the AVM that is running for the OpenEdge project. Deselect this option to have each new launch configuration run under a separate, dedicated AVM instance. You can edit this setting for specific configurations on the Main tab of the Configurations wizard.</td>
</tr>
<tr>
<td>Use selection in workspace for new launch configurations</td>
<td>If checked, causes new launch configurations to run the ABL program that is selected in the workspace or open in the ABL Editor. Deselect this option to have each new launch configuration run a specific program. You can edit this setting for specific configurations on the Main tab of the Configurations wizard.</td>
</tr>
</tbody>
</table>

See also
Launch configurations for running and debugging programs on page 551
Managing launch configurations on page 566
Launch Configuration Settings on page 589

Debugger dialogs

Debug Preferences dialog

The Debug Preferences dialog lets you enable the evaluation of properties with accessors and set the timeout value for the Debugger connection to the OpenEdge Client. This dialog appears when you select Window > Preferences > Progress OpenEdge > Debug.

You can set the following preferences:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable property evaluation</td>
<td>Enables the evaluation of properties with accessors. When you disable this preference, the actual value does not appear for the property in the Variables view. Instead, the message &quot;Property evaluation disabled&quot; appears. This preference is disabled by default.</td>
</tr>
</tbody>
</table>
Always select first matching file upon CRC mismatch:
Always selects the first matching file and ignores the rest, if there are one or more files with a similar name but none of them match with the CRC of the program being debugged.

Connection timeout (ms)
Sets the time-out used by the Debugger to connect to the OpenEdge Client.
The default value is 20000 ms.

See also
OpenEdge Debugger preferences on page 556
Setting Debugger preferences on page 569

Variable Details dialog
The Variable Details dialog displays information about a single data element. It is particularly useful for viewing information about the attributes and fields of HANDLE objects.
This dialog appears when you select an entry in the Variables view and select Variable Details from either the context (right-click) menu or the OpenEdge menu.
Information appears on one or more of four tabs: Attributes, Fields, Array, and Value. In the Value column (but not on the Value tab), in-place editing is supported where valid.
The following controls are available:

<table>
<thead>
<tr>
<th>View field</th>
<th>The name of the data element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>View command</td>
<td>To update the display after changing the entry in the View field.</td>
</tr>
<tr>
<td>Add Watch command</td>
<td>To add the current element to the list being monitored in the Expressions view.</td>
</tr>
<tr>
<td>Close command</td>
<td>To dismiss the dialog.</td>
</tr>
</tbody>
</table>

See also
Options for monitoring results on page 561
Using the Variable Details dialog on page 582
Using the Variables view on page 579
Expressions view on page 628

Select Columns dialog
The Select Columns dialog lets you choose which information appears in the Variables view:

<table>
<thead>
<tr>
<th>Name</th>
<th>The name of the variable.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declared Type</td>
<td>The way the type is defined.</td>
</tr>
</tbody>
</table>
### Value
The value of the variable.

### Actual Type
The type object associated with the handle variable (handle:TYPE) and the actual class for class variables. If the variable has not been initialized, the value shows as a question mark, indicating it is unknown.

This dialog appears when you click the drop-down menu in the **Variables** view, select **Layout**, and choose **Select Columns**. (You must also have chosen the **Show Columns** option for the **Select Columns** option to appear.)

**See also**
- Using the Variables view on page 579
- Variables view on page 626

### Set Value dialog
The **Set Value** dialog lets you change the value of a selected variable for the current debugging session without modifying the source code.

This dialog appears when you select an entry in the **Variables** view, right-click, and select **Change Value** from the context (right-click) menu.

**See also**
- Changing program code on page 586
- Variables view on page 626

### New Breakpoint dialog
The **New Breakpoint** dialog lets you define all types of breakpoints (**at-line**, **on-error**, **watchpoint**), including conditional breakpoints.

This dialog appears when you right-click in the **Breakpoints** view and select **Add Breakpoint**.

**See also**
- Breakpoints on page 560
- Setting breakpoints on page 573
- Breakpoints view on page 627

### Edit Breakpoint dialog
The **Edit Breakpoint** dialog lets change the definition of an existing breakpoint. You cannot change the breakpoint type (**at-line**, **on-error**, or **watchpoint**).

This dialog appears when you select an entry in the **Breakpoints** view, right-click, and select **Edit Breakpoint**.

**See also**
- Breakpoints on page 560
- Setting breakpoints on page 573
- Breakpoints view on page 627
**Add Watch Expression dialog**

The **Add Watch Expression** dialog lets you define a watch to be monitored in the **Expressions** view. This dialog appears when you right-click in the **Expressions** view and select **Add Watch Expression**.

Select or deselect the **Enabled** option to control whether monitoring is active for the watch. The expression appears in the **Expressions** view whether or not it is enabled.

**See also**
- Options for monitoring results on page 561
- Using watch expressions on page 583
- Expressions view on page 628

**Edit Watch Expression dialog**

The **Edit Watch Expression** dialog lets you change an existing watch expression. This dialog appears when you select an entry in the **Expressions** view, right-click, and select **Edit Watch Expression**.

Select or deselect the **Enabled** option to control whether monitoring is active for the watch. The expression appears in the **Expressions** view whether or not it is enabled.

**See also**
- Options for monitoring results on page 561
- Using watch expressions on page 583
- Expressions view on page 628

**Dynamic Object Monitoring dialog**

The **Dynamic Object Monitoring** dialog lets you turn on monitoring of selected types of dynamically created objects so that you can track their state in the **Dynamic Objects** view.

This dialog appears when you select **OpenEdge > Dynamic Object Monitoring** on the menu bar.

The following controls are available:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object type tree</strong></td>
<td>The expandable/collapsible list of dynamic object types that can be monitored, with check boxes for selecting.</td>
</tr>
<tr>
<td><strong>Start Monitoring command</strong></td>
<td>To turn on monitoring for the selected object types. This button changes to <strong>Restart Monitoring</strong> when you click it.</td>
</tr>
<tr>
<td><strong>Restart Monitoring command</strong></td>
<td>To restart or continue monitoring for the selected object types, clearing the current contents of the <strong>Dynamic Objects</strong> view.</td>
</tr>
<tr>
<td><strong>Close command</strong></td>
<td>To dismiss the dialog.</td>
</tr>
</tbody>
</table>

**See also**
- Options for monitoring results on page 561
- Using the Dynamic Objects view on page 585
Dynamic Objects view on page 629

Filter Objects dialog

The Filter Objects dialog lets you control which of the dynamic object types that are currently being monitored appear in the Dynamic Objects view.

This dialog appears when you right-click in the Dynamic Objects view and select Filter Objects from the context menu.

Information appears in the Dynamic Objects view about dynamic objects of the checked types.

See also
Options for monitoring results on page 561
Using the Dynamic Objects view on page 585
Dynamic Objects view on page 629

Select Process dialog

The Select Process dialog allows you to obtain the list of currently running OpenEdge processes on the same machines as the Progress Developer Studio for OpenEdge Debugger. To access this dialog, click Select Process on the Connection tab in the Progress External OpenEdge Application section of the Debug Configurations wizard. For more information, see Connection tab (Progress OpenEdge External AVM) on page 615.

See also
Attachable Debugger on page 557
Attaching the Debugger to an external AVM on page 572
Connection tab Progress OpenEdge External AVM on page 615

Add Source dialog

The Add Source dialog lets you add custom source lookup paths to the entries on the Source tab. To access this dialog, select Add on the Source tab. For more information, see Source tab (Progress Source tab OpenEdge AppServer and WebSpeed) on page 612 or Source tab (Progress OpenEdge External AVM) on page 616.

You can choose from the following containers to add to the source lookup path entries:

- File System Directory
- OpenEdge Project
- Workspace Folder

See also
Attachable Debugger on page 557
Attaching the Debugger to an external AVM on page 572
Suspending and disconnecting the Debugger from an attached AVM on page 588
Startup tab Progress OpenEdge AppServer and WebSpeed on page 605
Source tab Progress Source tab OpenEdge AppServer and WebSpeed on page 612
Edit Source Lookup Path dialog on page 624
Edit Source Lookup Path dialog

The Edit Source Lookup Path dialog allows you to add source lookup paths for the file being debugged. You can access this dialog by selecting the Edit Source Lookup Path option. This option displays in the ABL Editor when Progress Developer Studio for OpenEdge fails to find a matching source file in the specified source lookup paths during debugging.

The following options and controls are available on the Edit Source Lookup Path dialog:

| Source lookup path | Specifies one or more source lookup paths where Progress Developer Studio for OpenEdge searches for the source code of the program being executed by the attached AVM.  
|                    | The default source lookup path includes $DLC\src and all the workspace project folders in alphabetical order.  
| Note:             | Progress Developer Studio for OpenEdge searches for the source file in the source lookup paths in the order in which they are displayed.  
| Add Workspace Directory | Adds a directory from the current workspace.  
| Add External Directory | Adds a directory from the file system.  
| Edit | Changes the value of the current entry.  
| Remove | Deletes an entry from the list.  
| Move Up, Move Down | Changes the position in the list of a current entry, and thus changes the search order.  

Note: Any source lookup path updates on this dialog reflect in the Source tab of the corresponding debug launch configuration.

See also
Attachable Debugger on page 557
Attaching the Debugger to an external AVM on page 572
Suspending and disconnecting the Debugger from an attached AVM on page 588
Startup tab Progress OpenEdge AppServer and WebSpeed on page 605
Source tab Progress Source tab OpenEdge AppServer and WebSpeed on page 612

Debugger views

Debug view

The Debug view shows one or more session trees for each active Debugger session. It also displays all the running agents on the broker which is attached to the Progress Developer Studio for OpenEdge Debugger during remote debugging. It changes dynamically as agents are added or removed from the attached broker.
The **Debug** view also provides the **program execution commands** that you use to step through the code. The **Debug** perspective includes the **Debug** view by default. If the view is not open, select **Window > Show View > Debug** from the **Debugger** perspective to display it.

### Session tree Debug view

The session shows the contents of the ABL stack trace (lines marked with 📑) for the currently executing procedure in each active debugging session. In its initial state, the session tree shows the following entries:

- **Project name** [OpenEdge Application]
- **Source file name at localhost**
- **OpenEdge Runtime (suspended)**
- **Source file path and current line number**
- **OpenEdge-install-dir\bin\prowin32.exe** date and time

The stack trace changes dynamically as you step through the code, adding and removing entries that indicate the execution sequence of the current procedure. The current line appears at the top of the stack.

You can investigate the code associated with a line of the stack trace by clicking on the stack trace line. The **ABL Editor** window containing that code becomes active, and the **Variables** view, the **Expressions** view, and the **Dynamic Objects** view show data values as they are immediately preceding execution of the selected line.

See **Using the Debug view and stack trace** for more information.

### Program execution commands

The following commands are available on the **Debug** view toolbar and context (right-click) menu:

<table>
<thead>
<tr>
<th>Command</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step Into</strong></td>
<td>Execute the current line and continue until reaching the next executable statement, which may be in the current procedure, a subprocedure, or a trigger. That statement becomes the current line, and it is not executed until you continue. See also the information about garbage collection in <strong>Code stepping</strong>.</td>
</tr>
<tr>
<td><strong>Step Over</strong></td>
<td>Execute the current line and continue until reaching either a breakpoint, or the next executable statement in the same procedure. Thus, if the current line calls a subprocedure or causes a trigger to execute, the subprocedure or trigger is executed without interruption unless there is a breakpoint. In other words, step &quot;over,&quot; not &quot;into,&quot; the subprocedure or trigger.</td>
</tr>
<tr>
<td><strong>Step Return</strong></td>
<td>Execute the current line and continue either until reaching either a breakpoint, or until the current procedure ends and returns control to the calling procedure.</td>
</tr>
<tr>
<td><strong>Resume</strong></td>
<td>Execute the current line and continue without interruption until reaching either a breakpoint or the end of the program.</td>
</tr>
<tr>
<td><strong>Suspend</strong></td>
<td>Interrupt execution following completion of the current instruction and give control to the Debugger at the next executable statement. Also, suspend the attached AVM.</td>
</tr>
</tbody>
</table>
### Variables view

The **Variables** view shows current information about buffers, datasets, parameters, properties, variables, and temp-tables while program execution is suspended during a Debugger session. It is synchronized with the stack trace, so that you can examine values at different points of execution. The **Variables** view allows you to make non-persistent changes to values and also allows you to **evaluate properties**.

By **selecting the layout for the view**, you can choose how you want the information to appear and how much of it you want to see. The following information appears by default (although you can opt not to see these details by modifying the layout):

- **Name** - The name of the variable
- **Value** - The value of the variable

You can choose to see the following information in the layout as well:

- **Declared Type** - The way the type is defined.
- **Actual Type** - The type object associated with the handle variable (handle:TYPE) and the actual class for class variables. If the variable has not been initialized, the value shows as a question mark, indicating it is unknown.

You can toggle whether properties of a class type are displayed or not in the **Variables** view by working with the **Filter** option in the view's context menu. Settings are remembered through multiple Progress Developer Studio for OpenEdge sessions. The filters apply only to items in the **Variables** view and apply to all instances shown in the top level of the **Variables** view.

Handle variables, class variables, and properties of a class type are expandable in the **Variables** view, provided they have been initialized. Arrays, buffers, and temp-tables can be expanded even if they are not initialized.

Private data members and properties will be displayed for ABL objects. However, private data members and properties of .NET objects are not displayed.

The **Debug** perspective includes the **Variables** view by default. To display it if is not open, select **Window > Show View > Debug > Variables**.

The following commands are available:

<table>
<thead>
<tr>
<th>Command</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terminate</td>
<td>Interrupt execution and end the Debugger session.</td>
</tr>
<tr>
<td>Remove All Terminated</td>
<td>Clear the <strong>Debug</strong> view of session trees that are no longer active.</td>
</tr>
<tr>
<td>Launches</td>
<td></td>
</tr>
<tr>
<td>Disconnect</td>
<td>Disconnect from the attached AVM and stop receiving notifications from the running external AVM.</td>
</tr>
</tbody>
</table>

### See also
- Attachable Debugger on page 555
- Code-stepping on page 560
- Options for monitoring results on page 561
- Using the Debug view and stack trace on page 578
- Stepping through the code on page 577
To collapse the display of all hierarchical data elements in the list, leaving only the top-level entities visible.

To open a drop-down menu that provides options for customizing the appearance of the view. The Filter option lets you select the data element types that you want to include in the Variables view.

The drop-down menu provides these options:

<table>
<thead>
<tr>
<th>Layout</th>
<th>Lets you choose to display the information in the following ways:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Vertical View Orientation</td>
</tr>
<tr>
<td></td>
<td>• Horizontal View Orientation</td>
</tr>
<tr>
<td></td>
<td>• Variables View only</td>
</tr>
<tr>
<td></td>
<td>When a Debug session is selected in the Debug view, the Show Columns and Select Columns options are also available.</td>
</tr>
</tbody>
</table>

| Filter          | Show buffers, datasets, parameters, properties, variables, and/or temp-tables. |

See also
- Options for monitoring results on page 561
- Attachable Debugger on page 555
- Using the Variables view on page 579
- Using the Variable Details dialog on page 582
- Variable Details dialog on page 620

Breakpoints view

The Breakpoints view lists all breakpoints that are defined for the current workspace, including those set in procedures and projects that are not related to the application that you are debugging and for the files located outside the workspace. You can selectively enable or disable breakpoints for the current debugging session.

The Debug perspective includes the Breakpoints view by default. To display it if it is not open, select Window > Show View > Debug > Breakpoints.

Each entry label shows the definition of a breakpoint. If the breakpoint is at an include file reference or is conditional, that information is appended to the label.

The following commands are available:

| Remove Selected Breakpoints | To permanently delete currently selected breakpoints. |
| Remove All Breakpoints      | To permanently delete all breakpoints defined for the workspace. |
| Skip All Breakpoints        | To disable all breakpoints without deleting them. |

The following commands are available only if the breakpoints are grouped:
Expand All  
To expand all the items in the view.

Collapse All  
To collapse all the items in the view.

The drop-down menu provides these options:

<table>
<thead>
<tr>
<th>Group By</th>
<th>Breakpoints</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Breakpoint Types</td>
</tr>
<tr>
<td></td>
<td>Breakpoint Working Sets</td>
</tr>
<tr>
<td></td>
<td>Files</td>
</tr>
<tr>
<td></td>
<td>Projects</td>
</tr>
<tr>
<td></td>
<td>Resource Working Sets</td>
</tr>
<tr>
<td></td>
<td>Advanced</td>
</tr>
</tbody>
</table>

See also
- Options for controlling program execution on page 559
- Options for monitoring results on page 561
- The OpenEdge Debugger perspective on page 555
- Using Breakpoints on page 573
- New Breakpoint dialog on page 621
- edit Breakpoint dialog on page 621

Expressions view
The **Expressions** view lists the values of variables that you have explicitly selected by creating watch expressions. It is synchronized with the stack trace, so that you can examine values at different points of execution. The **Breakpoints** view, the **Variables** view, and the **Variable Details** dialog include facilities for creating watch expressions.

Class variables and properties of a class type are expandable in the **Expressions** view. Arrays, buffers, temp-tables, datasets, and handle variables can also be used in the **Expressions** view and are expandable.

An object reference or handle not initialized cannot be expanded. Arrays, buffers, and temp-tables can be expanded even if they have not been initialized.

Private data members and properties are displayed for ABL objects. However, private data members and properties of .NET objects are not displayed.

The **Debug** perspective includes the **Expressions** view by default. To display it if is not open, select Window > Show View > Expressions.

Each entry label shows the expression and its current value, "Unavailable," or "(disabled)."

The following label shows the expression and its current value, "Unavailable," or "(disabled)."

The following commands are available:

<table>
<thead>
<tr>
<th>Collapse All</th>
</tr>
</thead>
<tbody>
<tr>
<td>To collapse the display of all hierarchical data elements in the list, leaving only the top-level entities visible.</td>
</tr>
</tbody>
</table>
Remove currently selected watch expressions.

Remove All

To delete currently selected watch expressions.
To delete all watch expressions defined for the workspace.

The drop-down menu provides these options:

<table>
<thead>
<tr>
<th>Layout</th>
<th>Lets you choose to display the information in the following ways:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Vertical View Orientation</td>
</tr>
<tr>
<td></td>
<td>• Horizontal View Orientation</td>
</tr>
<tr>
<td></td>
<td>• Expression View Only</td>
</tr>
</tbody>
</table>

See also
Options for monitoring results on page 561
The OpenEdge Debugger perspective on page 555
Using watch expressions on page 583
Setting breakpoints on page 579
Using the Variable Details dialog on page 582
Add Watch Expression dialog on page 622
Edit Watch Expression dialog on page 622

Dynamic Objects view

The **Dynamic Objects** view lets you monitor the creation and destruction of dynamic object instances. You must explicitly start dynamic object monitoring for specific object types if you want to use this view.

The **Debug** perspective includes the **Dynamic Objects** view by default. To display it if it is not open, select either **Window > Show View > Dynamic Objects** or **Window > Show View > Other > Progress OpenEdge Debugger > Dynamic Objects**.

While the Debugger is monitoring active object instances, the following information is displayed in the **Dynamic Objects** view:

- The object type.
- The name of the procedure file in which the dynamic object was created.
- The line number at which the object was created.
- Any relevant attribute values (such as the value of the object's **Name** attribute) or other object-specific information.
- The path.

See also
Options for monitoring results on page 561
The OpenEdge Debugger perspective on page 555
Using the Dynamic Objects view on page 585
Dynamic Object Monitoring dialog on page 622
Add Watch Expression dialog on page 622
Introducing the ABL GUI Designer

The new improved seamless AppBuilder integration makes Developer Studio a one-stop solution for developing and maintaining ABL GUI applications.

The AppBuilder integration introduces a new design editor for designing AppBuilder GUI applications called ABL UI Designer. This design editor allows developers to draw widgets and generate source code, similar to the Visual Designer that is used to build GUI for .NET applications.

The following OpenEdge AppBuilder capabilities can be accessed from within the Developer Studio:

- OpenEdge AppBuilder perspective to build AppBuilder GUI applications within an OpenEdge project.
- AppBuilder options available as AppBuilder preferences and project properties in the Developer Studio. These options allow developers to control the behavior of AppBuilder and ABL UI Designer such as Dynamics, Editor, Grid Units, and Widget ID settings.
- ABL GUI applications are OpenEdge project-aware. That is, developer can set common properties for all the ABL GUI applications defined in an OpenEdge project using AppBuilder properties.
- AppBuilder templates and wizards to develop new ABL GUI applications. Stand-alone AppBuilder templates and wizards run embedded in Eclipse standard wizards from within the Developer Studio.
- Design ABL GUI applications using the ABL UI Designer editor which extends the native AppBuilder functionality such as dock-able widget palette, look and feel of Visual Designer editor, hierarchical widget Outline view, and synchronization between the AB UI Designer and ABL Editor.
- Improved ABL Editor to make sections of the AppBuilder-generated code read-only and foldable. This ensures that developers do not inadvertently modify the AppBuilder-generated code while editing the ABL application code is the ABL Editor.
- AppBuilder Section Editor to view specific block of code using the object nodes created in the Outline view. You can use the Section Editor Toggle button to enable or disable the Section Editor capability.
• Progress Dynamics features such as Open Dynamics Object, Register to Repository, Save Dynamics Object as Static, Save Static Object as Dynamic, and Open Associate Procedure.

• PRO*Tools utilities categorized into these three menu options: General PRO*Tools, AppBuilder PRO*Tools, and ADM2 PRO*Tools.

The OpenEdge AppBuilder can either run embedded in Progress Developer Studio for OpenEdge or as a standalone application.

For details, see the following topics:

• Concepts
• Tasks
• Reference

Concepts

This section includes the following topics:

OpenEdge AppBuilder perspective

The OpenEdge AppBuilder perspective includes various tools and views that you can use to build and develop AppBuilder applications in Progress Developer Studio for OpenEdge.

A new GUIDesigner editor gives Progress Developer Studio for OpenEdge the ability to design GUI for ABL user interfaces using the native AppBuilder functionality to draw the widgets and to parse and generate source code, similar to how the Visual Designer uses Visual Studio to build GUI for .Net user interfaces.

The GUI designer will reuse common behavior like the ability to jump back and forth between visual layout and source editing.

The GUI Designer has a new dockable Palette that extends Eclipse’s existing Palette and is populated with components from the AppBuilder’s customizable configuration files.

See also

Introducing the ABL GUI Designer on page 631
Accessing the AppBuilder perspective on page 641
AppBuilder perspective components on page 676

Benefits and limitations of GUI Designer

There are certain benefits and limitations of GUI Designer (AppBuilder in Progress Developer Studio for OpenEdge environment) over standalone AppBuilder, which are as follows:

• The GUI Designer project properties are changed as part of the integration. The GUI Designer project properties are stored as OpenEdge project properties. Some of the .ini or registry settings that are used by standalone AppBuilder is ignored when running in Progress Developer Studio for OpenEdge (Developer Studio). See Designer project properties.

• The GUI Designer preferences are changed as part of the integration. The GUI Designer preferences are stored as OpenEdge preferences and are workspace level settings. The .ini or registry settings that are
used by standalone AppBuilder are ignored when running in OpenEdge. See AppBuilder preferences on page 677.

Note: The Progress Developer Studio for OpenEdge stores the GUI Designer preferences settings in Eclipse preference store, and sends these settings to standalone AppBuilder to synchronize the preferences settings.

• You can open multiple GUI Designer procedure file (.w) windows. See Working with ABL procedure files on page 646.

• Cue Cards appear as a view in AppBuilder perspective in Developer Studio. In standalone AppBuilder, you can control the display of Cue Cards from the Preferences page. See ABL Cue Cards view on page 816.

• Recent Messages feature is not available with GUI Designer in Developer Studio. You can access this feature from the Help menu in the standalone AppBuilder.

• The Toggle on/off to change window on-top setting option on the Dynamic Properties window does not work in Developer Studio environment. This option only works when the Dynamics Properties window is opened in the standalone AppBuilder environment.

Note: The Dynamic Properties window opens and runs in a window outside of the Progress Developer Studio Eclipse framework. Online help is available from the utility's Help button or F1 key

• Non Visual AppBuilder files, including Web Objects such as HTML Mapping objects will open in the ABL UI Designer in the same tree view and have the same behavior as in standalone AppBuilder, with the exception that clicking on the Code Section nodes is procedure/function nodes. There is no wizard support to create AppBuilder web objects in Developer Studio.

See also
OpenEdge AppBuilder perspective on page 632

Code analysis in GUI Designer

The &ANALYZE directives are commands understood by the OpenEdge Analyzer, a component of the OpenEdge Compiler. The Analyzer's main purpose is to make information about static objects available to the AppBuilder. The activities of the Analyzer are completely transparent, except for the &ANALYZE directives that appear in your code. You can use the GUI Designer effectively without knowing anything about the Analyzer, but if you want to know more about the directives appearing in your code, continue reading.

There are two main &ANALYZE directives: &ANALYZE-SUSPEND and &ANALYZE-RESUME. The &ANALYZE-SUSPEND directive causes the Analyzer to suspend analysis of ABL code. The &ANALYZER-RESUME directive causes the Analyzer to resume analysis.

The Analyzer performs its work when the GUI Designer opens a .w file. When the GUI Designer opens a .w file, the Analyzer builds a temporary file that contains:

• A summary table of information about the static objects created with the GUI Designer. User-supplied object definitions, whether static or dynamic, are not analyzed.

• Exact copies of any user-supplied code (triggers, variable definitions, etc.).

• Exact copies of any AppBuilder -generated code that is not analyzed (such as CREATE WINDOW statements).
The GUI Designer then reads contents of the temporary file into memory, storing the information in temporary tables.

The summary table is the Analyzer's most important output and is the Analyzer's main purpose. This table contains all of the relevant information about the .w file's static objects, including their dimensions. The format of the summary table is one that the GUI Designer can easily parse. Using information in this table, the GUI Designer creates design-time representations of the static objects. These representations are what you physically manipulate within the design window.

The Analyzer cannot provide summary information about dynamic objects. Dynamic objects are created at run time, not defined at compile time. The GUI Designer must examine all unanalyzed code, extracting information it requires to create design windows and dialog boxes.

**See also**

Coding conventions to generate AppBuilder code on page 634
Viewing and editing the source code on page 655

---

**Coding conventions to generate AppBuilder code**

The following describes the conventions the GUI Designer uses when generating code:

**Absolute Positioning**

The GUI Designer uses absolute positioning in all of its generated ABL code. You must take this into consideration before building your application.

ABL offers two alternative syntaxes for positioning objects within frames:

- This syntax allows you to place an object at a specific location within a frame. This syntax supports absolute positioning. That is, you position objects at a specific X or Y location, or at a specific COLUMN or ROW.
- This syntax supports relative positioning. That is, you place objects relative to each other in a frame using offsets.

For more information about these two types of syntax, see OpenEdge Development: ABL Handbook and OpenEdge Development: Programming Interfaces.

**Absolute Field References**

When referencing database fields, the GUI Designer does not fully qualify the database field names with their respective database names. If you want database field names to be absolute (specifying the database name, the table name, and the field name), you can override the default by selecting the Qualify database fields with database name check box in the AppBuilder Properties page.

**Character Units**

To specify object dimensions and positions, the GUI Designer uses character units. You can override this default on an object-by-object basis within the object's advanced Property Sheet. The alternative to character units is PIXELS.

**See also**

Code analysis in GUI Designer on page 633
Viewing and editing the source code on page 655
**Read-only code sections of AppBuilder-generated code**

You can make the sections of the AppBuilder-generated code read-only and foldable in the ABL Editor. This ensures that users do not modify the AppBuilder-generated code while editing the ABL procedure (.w) file in the ABL Editor. Editing the AppBuilder-generated code might make the AppBuilder procedure (.w) file corrupt, and display errors when trying to open in the ABL GUI Designer.

The default display color for the read-only code is gray. You can set the display options of AppBuilder-generated code in the ABL Editor using the Editor preferences page. See Editor preferences on page 678.

---

**Note:** The Find/Replace option will not work with the read-only or auto-generated code of an AppBuilder procedure file. You cannot rename an internal procedure or function name which is part of the auto-generated or read-only code section.

---

There are two main &ANALYZE directives in an AppBuilder (.w) file: &ANALYZE-SUSPEND and &ANALYZE-RESUME. The &ANALYZE-SUSPEND directive causes the Analyzer to suspend analysis of ABL code. The &ANALYZE-RESUME directive causes the Analyzer to resume analysis. See Code analysis in GUI Designer on page 633.

The identifiers that define the read-only rules are the &ANALYZE-SUSPEND and &ANALYZE-RESUME statement directives. When an ABL procedure (.w) file is opened in the GUI Designer, the &ANALYZE-SUSPEND and &ANALYZE-RESUME directives code are parsed to mark the code sections within these directives as read-only and foldable in the ABL Editor.

Here is a list of sample illustrations with the code snippets marking the code section or block within the &ANALYZE directives as read-only or editable:

- The following code snippet specifies that the code generated inside the &ANALYZE block is read-only except the definition section which is editable:

```abl
&ANALYZE-SUSPEND_UIB-CODE-BLOCK_CUSTOM_DEFINITIONS_WHIN

/* File: 
   Description: from cntnrwin.w - ADM SmartWindow Template 
   Input Parameters: 

   /* This .W file was created with the Progress AB. 
   //________________________________________________________________________ 
   /* Create an unnamed pool to store all the widgets created 
   by this procedure. */
   CREATE WIDGET-POOL. 

   /****************************************** Definitions ********************** */
   /* Local Variable Definitions -- */
   [src/adm2/widgetsetc.w]

   /_UIB-CODE-BLOCK-END 
&ANALYZE-RESUME
```

- Any code which is not surrounded by &ANALYZE-SUSPEND and &ANALYZE-RESUME is read-only.
- An &ANALYZE-SUSPEND statement which does not have _UIB-CODE-BLOCK or _QUERY-BLOCK as the second token is read-only.
- An &ANALYZE-SUSPEND _UIB-CODE-BLOCK_CUSTOM_DEFINITIONS block is editable.
- The following code snippet specifies how triggers and the main block are editable:
- An &ANALYZE-SUSPEND _UIB-CODE-BLOCK_CUSTOM_DEFINITIONS block is not read-only.
- An &ANALYZE-SUSPEND _UIB-CODE-BLOCK with _CONTROL as the third token is not read-only, but the first line inside the block is read-only.
• The following code snippet specifies how procedures can be both read-only and editable, and the functions as editable. Note that the END PROCEDURE and END FUNCTION are editable:

```abl
&ANALYZE-SUSPEND _UID-CODE-BLOCK _PROCEDURE exitObject wWin
PROCEDURE exitObject
    
    // Purpose: Window-specific override of this procedure which destroys its contents and itself.
    // Notes: 
    // APPLY "CLOSE";U TO THIS-PROCEDURE.
    RETURN.
END PROCEDURE.

&ANALYZE-SUSPEND _UID-CODE-BLOCK _PROCEDURE getCurrent wWin
FUNCTION getCurRent RETURNS [CHARACTER]
    (oMode as character)
    
    // Purpose: A function to get the current record of the field.
    // Notes: 
    // RETURN "current". // Function return value. 
END FUNCTION.

&ANALYZE-SUSPEND _UIB-CODE-BLOCK _PROCEDURE with five tokens is not read-only.

• An &ANALYZE-SUSPEND _UIB-CODE-BLOCK _PROCEDURE with six tokens is read-only, unless the sixth token is _FREEFORM.

• An &ANALYZE-SUSPEND _UIB-CODE-BLOCK _FUNCTION is editable. The data-type of the function must be editable.

• When you add Extended Features (XFTR) block to a procedure or template, then the code becomes editable.

• The following code snippet specifies how free form query elements inside a &ANALYZE-SUSPEND _QUERY-BLOCK section are editable:

```
• An &ANALYZE-SUSPEND _UIB-CODE-BLOCK is editable between _START_FREEFORM and _END_FREEFORM or _START_FREEFORM_DEFINE and _END_FREEFORM_DEFINE tokens. These tokens are inside of comments.

• The following code snippet specifies how a browser definition for a free form query is editable, allowing you to edit the field lists:

```plaintext
DEFINE BROWSE BROWSE-4
  &ANALYZE-SUSPEND _UIB-CODE-BLOCK DISPLAY-FIELDS BROWSE-4
  &START_FREEFORM
  QUERY BROWSE-4 NO-LOCK DISPLAY
  Customer.Address FORMAT "(##)";U
  Customer.Address2 FORMAT "(##)";U
  Customer.City FORMAT "(##)";U
  Customer.Comments FORMAT "(00)";U
  Customer.Country FORMAT "(##)";U
  Customer.CreditLimit FORMAT "(##)", "", "", "", >"";U
  CUSTOMER.CustNum FORMAT "(>>>9)";U
  ENABLE
  Customer.Address
  Customer.Address2
  Customer.Balance
  Customer.City
  Customer.Comments
  Customer.Country
  Customer.CreditLimit
  CUSTOMER.CustNum
  /* _UIB-CODE-BLOCK-END */
  &ANALYZE-RESUME
```

• An &ANALYZE-SUSPEND _UIB-CODE-BLOCK _DISPLAY-FIELDS section is editable when the sixth token is _FREEFORM.

See also
- Using code folding on page 633
- Editor preferences on page 678

About SmartObjects

SmartObjects are external procedures that encapsulate standard UI and functionality. The set of SmartObjects provided with the Application Development Model (ADM) are useful for typical database applications. For example, most database applications view data, so the ADM provides a SmartV8Viewer object. A SmartV8Viewer's main purpose is to display data.

Each of these SmartObjects represents a component part of a typical database application.

Each SmartObject must know how to interact with other SmartObjects. For example, the SmartV8Query and SmartV8Viewer work together in predefined ways. The SmartV8Query tells the SmartV8Viewer what records to display on the application screen. Each of these objects expects the other to behave in specific ways; each has responsibilities to the other.

As a developer, you can quickly assemble SmartObjects together to build many different application screens. If a SmartObject behaves in way that you don't like, you can customize it or override its default behaviors. If the set of available SmartObjects does not suit your needs, you can create new SmartObjects, giving them different behaviors better suited to the role you assign them in your application.

Each SmartObject has information (attributes) that it stores internally. A given SmartObject accesses this information by running internal procedures in another SmartObject.

Each SmartObject is capable of performing specific actions. These actions are represented as internal procedures. Conceptually, there are two types of actions that a SmartObject can perform: actions made available to other SmartObjects (methods), and default actions that govern how the SmartObjects behaves at critical moments in an application (ADM events).
See also
Common capabilities of SmartObjects on page 640
SmartObject controls on page 684
Open SmartObject dialog on page 788

SmartObject instance

SmartObject instance is a copy of a SmartObject master running in memory. Many SmartObject instances can be created from a single SmartObject master file.

See also
About SmartObjects on page 637
Inserting SmartObjects on page 669

SmartObject master

SmartObject master is a SmartObject with specific data and UI. Generally, you create a SmartObject master from a SmartObject template. You can add data and UI elements to the template and then save the file. The saved file then becomes a SmartObject master. You can use a master as a template for other masters, which in turn add more specific details.

See also
About SmartObjects on page 637
Inserting SmartObjects on page 669

Paging in SmartObjects

Every SmartContainer (SmartWindow, SmartFrame, and SmartDialog) gives you the ability to organize sets of SmartObjects into logical pages. The primary purpose of the paging mechanism is to give you control over when these sets of objects are initialized and displayed.

This allows you to make the best use of limited screen space, and to keep the attention focused by limiting the amount of information displayed at one time.

The SmartFolder provides a convenient way to reserve screen space and control the display of logical pages. The GUI Designer also allows you to manage pages in other ways. For example, you can use the paging mechanism to create multiple-window applications.

Special pages

There are three page designations that have special meaning in the paging system:

<table>
<thead>
<tr>
<th>Page</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Page 0 is the default page in every SmartContainer. If you create a SmartContainer and add objects to it without changing any of the paging settings, then all the objects are on Page 0. Also, any basic objects (widgets) you add directly to a SmartContainer are placed on Page 0 regardless of the Design Page setting. Objects placed on Page 0 are always displayed in addition to any objects on any other pages that have been selected for display.</td>
</tr>
</tbody>
</table>
### Startup page
You have the option of designating one page, in addition to Page 0, to be displayed at the time the SmartContainer is run. This is called the Startup Page.

Page 0 is the default Startup Page. Unless you explicitly specify a non-zero page number as the Startup Page, only the objects placed on Page 0 will display when you run the SmartContainer.

For example, if you are using a SmartFolder to organize the viewing and hiding of pages, you would typically designate Page 1 (which always corresponds to the leftmost folder tab) as the Startup Page. If you leave the Startup page set to Page 0 when you are using a SmartFolder, then the area within the folder object will be blank when the application starts up.

You set the Startup Page in the Pages dialog box, which you invoke from the Procedure Settings dialog.

### Design page
This is the page you are working in at any given time while you are building the application. As the name suggests, the Design Page setting is relevant only in design mode. It is ignored when the SmartContainer is running.

The Design Page setting is indicated in the status bar of the GUI Designer window. You can set the Design Page by double-clicking the Page Number field in the GUI Designer main window, or by choosing the Design button in the Pages dialog box.

### Paging system limited to SmartObjects
It is important to note that only SmartObjects can be placed on pages other than Page 0.

Only SmartObjects are capable of responding to SmartLinks, and a SmartContainer controls the paging system by means of a set of special SmartLinks. When you assign a SmartObject to Page n, the GUI Designer automatically creates a SmartLink of type Pagen from the SmartContainer to the SmartObject. For more information see, About SmartObjects on page 637

### Layout
Layout is a collection of basic objects, SmartObject instances, and associated attribute settings. These attribute settings determine how the layout appears when you run the application. Each layout has a name and an associated run-time expression.

When you run a procedure file, it determines whether to use a layout by evaluating the layout's run-time expression. If the expression evaluates to TRUE, the file uses that layout. When the file determines that it uses a particular layout, the file executes a CASE statement entry for that layout. This CASE statement contains all of the attribute settings assigned for the layout.

At design time, a layout is visually manifested in the design window. You can switch between layouts in the design window, and the AppBuilder alters the appearance of the design window to match the characteristics of the new layout.

### About Method Library
The Method Library functionality allows you to add one or more include files into the generated code. The include file is inserted after all the definitions, and before any procedures and functions. Thus, allowing you to use include files that references the definitions or uses the preprocessors generated by the AppBuilder.
The term Method Libraries originally refers to the special type of include files that contain the methods for the various type of components of the ADM1 (Application Development Model) framework. The method libraries provide code reuse and ensure that each instance of an ADM1 class have the same logic being compiled into the r-code.

The ADM2 also uses the method libraries, but in this case the reusable code and class hierarchy are implemented in super-procedures. The method library defines the class of the instance, but contain very few methods, instead they initiate and bootstrap the class super procedure and super procedure hierarchy to the instance. They also include some variable definitions and preprocessors directives to control what to include in the compilation unit. But, the preprocessors generated by the GUI Designer are rarely used as compiler directives, but typically passed to the super procedures as property values.

**See also**
- Adding method library reference on page 671
- Modifying method library reference on page 671
- Method Libraries dialog on page 780

### Common capabilities of SmartObjects

All SmartObjects have common capabilities which allow them to interact in an application. All SmartObjects can:

- Initialize and destroy themselves
- Get and set attributes
- Add and remove SmartLinks
- Communicate using the Application Data Models (ADM) standard communication interface (mainly using the notify and dispatch methods)

These capabilities provide a standard interface for creating and destroying SmartObject instances and for making SmartObjects interact. Unless a procedure object has these capabilities, it is not a SmartObject.

**See also**
- About SmartObjects on page 637
- Open SmartObject dialog on page 788

### About Progress Dynamics

The Progress Dynamics framework is a comprehensive, repository-based environment for Progress Developer Studio for OpenEdge users who are building new applications, or who need to migrate existing applications to take advantage of new OpenEdge technologies.

A Progress Dynamics application is built up of many components of different types. Some of these provide support for the user interface, such as windows, browsers, viewers, toolbars, and tab folders. Others are procedural objects that define the business logic of an application. Many application components are dynamic or data-driven objects, created at run time using data in the Dynamics Repository. Progress Dynamics is to some extent based on the OpenEdge Application Development Model (ADM) and OpenEdge SmartObjects™, which provide a basis for defining and combining standard components. Progress Dynamics extends the ADM and adds even more capabilities to the OpenEdge development environment.
The Progress Dynamics Repository stores data for a wide variety of purposes. Most application objects are represented as data stored in the Repository, so you do not need to create, compile, deploy, or maintain the source code for them. The framework includes programs that render these objects at run time from the data in the Repository.

Progress Dynamics is designed to run in a distributed, n-tier environment, with the visual portion of the application running in either an ABL (Advanced Business Language) client session or some other client type, without a local database connection. Business logic runs on one or many AppServers for OpenEdge applications where you can locate the repository database and the application database, maximizing efficiency of access to the database.

See also
- Setting Progress Dynamics projects on page 663
- Opening objects from the Repository on page 664
- Progress Dynamics menu options on page 690
- Progress Dynamic dialogs on page 808
- Dynamic objects on page 686

About custom object file

An external text file or files in which you define custom objects using a specific syntax. Custom object files normally have a .cst extension in their filenames. OpenEdge includes two custom object files for you to use: progress.cst and smart.cst. Both files reside in DLC/src/template.

Custom object files let you control how three AppBuilder interface components ¾ the Object Palette, New dialog box, and Choose dialog box ¾ present objects to the user for selection. By editing a custom object file or by creating your own, you can prepare the presentation of your own set of objects.

See also
- Working with custom object files on page 672

Tasks

Accessing the AppBuilder perspective

To open the OpenEdge AppBuilder perspective:

1. Choose one:
   - Click Open Perspective on the Workbench toolbar.
   - Select Window > Open Perspective menu.

2. Select OpenEdge AppBuilder, if it is listed. If it is not listed, select Other. Select OpenEdge AppBuilder from the Open Perspective dialog, and then click OK.

   Note: To view the OpenEdge AppBuilder perspective if it is open but not in view, click OpenEdge AppBuilder from the Workbench toolbar.
Setting AppBuilder preferences

Use preference settings to modify the behavior of the GUI Designer in Progress Developer Studio for OpenEdge.

To modify the behavior of the GUI Designer:

1. Select **Window > Preferences**. The **Preferences** window appears.
2. Select the **Progress OpenEdge > AppBuilder** node. The **AppBuilder options** page appears with settings for behavior when double-clicking an object, default function data type, and display of Progress Advisors. See also **AppBuilder options**.
3. Expand the **AppBuilder** node to select from the following pages:
   - **Editor** - Options for how AppBuilder-generated code must appear in the ABL Editor
   - **Grid Units** - Options to set grid display options of the designer window
   - **Widget ID** - Options to set Widget ID options
4. Once you have made changes to the AppBuilder preference settings, click **Apply** to save your changes.

**Note:** Click **Restore Defaults** to reset AppBuilder preferences to their default values.

Setting AppBuilder properties

You can use the AppBuilder Properties page to you to set the behavior of the ABL GUI procedure (.w) files in the currently selected OpenEdge or Dynamics project.

To set the AppBuilder project properties:

1. In the Project Explorer view, right-click on your project, and then select **Properties** from the right-click context menu. The **Properties** page appears.
2. Select **Progress OpenEdge > AppBuilder**. The **AppBuilder Properties** page appears. The AppBuilder Properties page provides the following:
   - Dynamics settings
   - General information related to database
• Character Terminal Simulator color

3. Make the changes as required, and click **OK**.

**See also**

- Accessing the AppBuilder perspective on page 641
- Setting AppBuilder preferences on page 642
- Setting Progress Dynamics projects on page 663

## Creating an AppBuilder project

To create an OpenEdge project specialized for AppBuilder application development:

1. Select **File > New > OpenEdge Project** from the Progress Developer Studio for OpenEdge main menu bar or the Project Explorer view context (right-click) menu. The New OpenEdge Project wizard opens and displays the Create an OpenEdge Project page on page 105.

2. Type a name in the **Project name** field.
   
   The root folder for the new project is created in your current Eclipse workspace, unless you specify a different location.

3. To choose a different location for the project, click and clear the **Use default** check box. Then either browse to, or enter the path to the desired location.

4. Select **Client** from the left section.

5. Select the **Desktop Application** option button and select the **AppBuilder** option button.

6. Click **Next**. The Select AVM and layout options page on page 107 appears.

7. Specify the OpenEdge AVM options and the Project layout options (or leave the default settings) and click **Next**. The Define PROPATH page on page 109 appears.

8. Leave the default settings or modify the PROPATH entries to be used by the project and click **Next**. The Select database connections page on page 109 appears.

9. Select the database connections to be used by the project and click **Finish**.

You will see your project and its resources in the **Project Explorer** view.

**See also**

- Project types in Progress Developer Studio for OpenEdge on page 86
- Creating a new OpenEdge project
- Create an OpenEdge Project page on page 105
- Project Facets page on page 111
- Select AVM and layout options page on page 107
- Define PROPATH page on page 109
- Select database connections page on page 109
Working with AppBuilder procedure files

The ABL UI Designer editor is the default design editor and ABL Editor is the default source editor for the valid procedure .w files. The valid .w files mean the procedure files which are created using the standalone AppBuilder or GUI Designer using OpenEdge AppBuilder perspective in Progress Developer Studio for OpenEdge.

**Note:** You must be in the OpenEdge AppBuilder perspective to work with the .w files created in the GUI Designer or standalone AppBuilder application.

The procedure .w files created with the GUI Designer or standalone AppBuilder are indicated with an AppBuilder icon in the Project Explorer view.

**Note:** There may be .w files in an OpenEdge project that are not created using the GUI Designer or standalone AppBuilder. Such .w files do not open with the GUI Designer editor, instead open with the ABL Text editor.

Creating ABL GUI procedure files

The **New ABL UI Design** wizard allows you to create ABL GUI procedure files in your OpenEdge project. These files have the file extension as .w.

To create a new ABL GUI procedure (.w) file:

1. Do one of the following:
   - From the *File > New* New menu, select *Other*.
   - In the Project Explorer view, right-click a project and then select *New > Other*.

   The **Select a Wizard** dialog appears.

2. In the **Wizards** field, type ABL UI Design.

   **Note:** The ABL UI Design type is present under the Progress OpenEdge > AppBuilder node.

3. Click **Next**. The **New ABL UI Design** wizard appears.

4. By default, the **Container** field displays the selected project under which the new file is created. If you want to add the new procedure file to another project, do the following:
   a) Click **Browse** to open the **Folder Selection** dialog.
   b) From the **Select new file container** list, select a project.
   c) Click **OK**.

5. Do one of the following:
   - From the **Object Type** list, select a container or SmartObject type.

   **Note:** If you select a non-supported template, a warning message appears stating: Selected template is not supported in Progress Developer Studio. The wizard will be started outside of the workbench when you press **Finish**. Go to step 8 on page 645.
• If you want to create the new ABL procedure file using a template, then click Template: the Choose Other Template dialog appears. The selected template is added to a Template node. See Choose Other Template dialog.

The Template field displays the relative path and the Description field provides a description of the object type you select from the list.

6. In the File name field, enter a name for the ABL GUI procedure file or accept the default name.

7. Do one of the following:

• If you have selected a regular container template, go to 8 on page 645.
• If you have selected a SmartObject template, click Next and follow the wizard.

8. Click Finish.

This creates the ABL GUI procedure file and adds it to the selected OpenEdge project.

See also
Working with AppBuilder procedure files on page 644
Running ABL procedure files on page 670
Choose Template dialog on page 756

Opening ABL GUI procedure files
You can open an existing ABL GUI procedure (.w) file with the ABL UI Designer or ABL Editor.

To open an ABL GUI procedure (.w) file:

1. In the Project Explorer view, right-click a procedure file name and select Open With from the context menu.
2. Choose one:

• OpenEdge ABL Editor to display the source code.
• ABL UI Designer to display the design editor.

Alternatively, you can simply double-click the procedure file in the Project Explorer view. Progress Developer Studio for OpenEdge opens the procedure file either in an ABL UI Designer editing window or in the ABL Editor, whichever you last selected when opening the file with the Open With command. If you have not previously used Open With for the file, by default it opens in the ABL UI Designer.

Note: You cannot open multiple instances of the same file in the ABL UI Designer.

See also
Creating ABL GUI procedure files on page 644
Running ABL procedure files on page 670

Editing ABL procedure files
A .w file opened in the AppBuilder is displayed graphically in the GUI Designer design canvas. In addition, the code within the .w file is displayed in the ABL editor. The ABL editor allows full editing of the valid .w file. The textual and graphical views of the .w file are synchronized.
A modified marker * appears on both the views when you make a change in any of the view. The changes are synchronized when you switch views, and when you save the file. Thus, changes that you make in one view are immediately reflected in the other.

Synchronization is not performed immediately. The synchronization is performed when:

- Focus changes from the design window to view source editor.
- The ABL Editor loses focus.

It is possible that an analyzer error or a syntax error in the ABL Editor might make synchronization with the AppBuilder impossible. In this case, the window corresponding to the .w file in the AppBuilder closes. It is no longer valid, and cannot be modified. However, you can still use the ABL Editor to edit the .w file.

Once the file in the ABL Code Editor is updated, and synchronization takes place, the AppBuilder tries to load the .w file and the window opens if there are no errors.

The Edit > Undo feature or the local history feature can be used to recover a version of the file that will not cause an error while loading in the AppBuilder.

See also
- Creating ABL GUI procedure files on page 644
- Viewing and editing the source code on page 655

Working with ABL procedure files

The GUI Designer editor is the default editor for the procedure .w files created with the GUI Designer or standalone AppBuilder application. This editor is used to edit the ABL code of these valid .w files.

Note: You must be in the OpenEdge AppBuilder perspective to work with the .w files created in the GUI Designer or standalone AppBuilder application.

The .w files created with the GUI Designer or standalone AppBuilder are indicated with an AppBuilder icon in the Project Explorer view.

There can be .w files in an OpenEdge project that are not created using the GUI Designer or standalone AppBuilder. Such .w files do not open with the GUI Designer editor, instead open with only the ABL Text editor.

Opening .w files

Open .w files by using one of the following methods:

- Drag and drop the .w file from the Project Explorer view onto the GUI Designer window.
- Double-click on a .w file in the Project Explorer view.
- Choose File > Open from the main menu in the OpenEdge AppBuilder perspective.

See Viewing and editing the source code on page 655.

Editing .w files

A .w file opened in the AppBuilder is displayed graphically in the GUI Designer design canvas. In addition, the code within the .w file is displayed below the design canvas in the GUI Designer editor. The editor allows full editing of the .w file. The textual and graphical views of the .w file are synchronized. A change in one view causes a change in the other view.

Synchronization is not performed immediately. The synchronization is performed when:

- Focus changes from the AppBuilder to the ABL Editor.
• The ABL Editor loses focus.

It is possible that an analyzer error or a syntax error in the ABL Editor might make synchronization with the AppBuilder impossible. In this case, the window corresponding to the .w file in the AppBuilder closes. It is no longer valid, and cannot be modified. However, you can still use the ABL Editor to edit the .w file.

Once the file in the ABL Code Editor is updated, and synchronization takes place, the AppBuilder tries to load the .w file and the window opens if there are no errors.

The Edit > Undo feature or the local history feature can be used to recover a version of the file that will not cause an error while loading in the AppBuilder.

Note: Currently, templates for adding procedures, functions and triggers to .w files are not available. However, the .w files can be edited as long as the new code follows the structure of a .w file. In addition, there is an Insert > Trigger option available from the Edit menu in AppBuilder.

Saving .w files

You can save a file opened in the AppBuilder by selecting File > Save or File > Save As from the AppBuilder menu bar. If the file is part of a project, the Save and Save As operations map to the Save and Save As options in Progress Developer Studio for OpenEdge. You can also use the Save and Save As options directly from within Progress Developer Studio.

If a file opened in the AppBuilder is not a project file, the Save and Save As options can be used to save the file into the project directory.

Running a .w file

You can run a .w file by clicking the Run button on the AppBuilder main window or on the Progress Developer Studio for OpenEdge toolbar.

When you use the AppBuilder Run button, AppBuilder runs whatever file has focus in the design window. It runs the file just as you see it in the design window, even if the file has not been saved.

When you select Run from the Progress Developer Studio for OpenEdge toolbar, Progress Developer Studio runs the last saved version of whatever file has focus in the code editor. To run a file from the Progress Developer Studio toolbar that was changed in the design window, you have to save the file in the code editor.

Closing .w files

A project file opened in the AppBuilder can be closed either from the AppBuilder or from the ABL Editor.

Working with objects

Accessing property sheets

There are two basic ways to access the Property Sheet dialog for a selected object:

• Double-click the selected object. You must set the AppBuilder preferences on page 677 to open the Property Sheet dialog on double-clicking an object.

• Select an object in the design window, and then click Property Sheet icon on the toolbar.

For more information, see Property Sheet dialogs on page 702
**Using the Undo and Redo options**

You can undo any editing changes that you perform on the design window, and you can redo any changes that you have previously undone.

To undo a move, resize, align, or delete performed on an object in the design window, select **Undo** from the **Edit** menu or press Ctrl+Z. To redo an action, select **Redo** from the **Edit** menu or press Ctrl+Y.

The undo or redo changes are specific to the current design window. To undo or redo the actions, you must have focus on the design window.

**Note:** Only editing actions that have occurred since the most recent save operation can be undone and redone; you cannot use **Undo** to return to a state earlier than the latest saved version of the file. To revert to an earlier saved version, right-click the class file in the Project Explorer view and select **Replace With > Local History** from the context menu.

For more information, see Choosing objects for the design window

**Viewing and editing code of an object**

You might want to view the code section of an object instead of the complete code of the (.w) procedure file, when working with objects on the design window. This will be useful when you want to quickly add or edit a function, procedure, or trigger to an object.

**Note:** You cannot edit the auto-generated code of an AppBuilder file. The auto-generated code is read-only. See Read-only code sections of AppBuilder-generated code on page 635.

To view the code section of an object, double-click the object on the design window. The source view opens with the focus to the default trigger or event defined for the selected object. You can edit it.

If there is no trigger defined for an object, a default trigger section is created. You can define a trigger for an object.

**Note:** You must set the preferences to **Edit source code** in the **AppBuilder Preferences** page to open source view when you double-click an object. See AppBuilder preferences on page 677.

Alternatively, you can use the Outline view to go to specific places in the code. See Outline view on page 813.

**See also**

- Viewing and editing the source code on page 655
- Renaming an internal procedure or function on page 656
- Using the Find/Replace dialog on page 658

**Choosing objects for the design window**

This section explains how to populate an ABL UI Designer with objects.

**Adding objects to the design window**

**Placing objects on the design window**
There are several ways to place the objects from the Object Palette onto the design window:

- **Clicking** – Select an object in the Palette and click once on the design window at the required position to place an instance of the object as its default size.

- **Double-clicking** – Double-click an object on the Palette to add an instance of the object at its default size. The new object overlaps the last object that you operated on, if applicable, or appears at the top left corner of the design window. You will need to click Pointer icon in the Palette to be able to select other object in the Palette.

- **Drawing** – Select a basic object or OCX control on the Palette, point to the desired position on the design window, and draw a rectangular outline of the size and shape you want the object to have. When you release the mouse button, an instance of the object appears, occupying the outlined area.

- **Copying/cutting and pasting** – Click the pointer tool on the Palette to activate selection mode. Then select one or more objects on the design window, copy or cut them, and paste new instances onto the design window. The Copy (Ctrl+C), Cut (Ctrl+X), and Paste (Ctrl+V) commands are available on the **Edit** menu. This option only works with basic objects and OCX controls.

**Placing controls inside a container control**

In general, when you place an object on the design window, the object becomes a child object of the design window. Certain controls (for example, the Frame object), however, can contain other objects; in such cases, the Frame object is the parent of the child objects placed within it.

**Note:** Deleting the parent object deletes all its child objects placed within it.

**Placing database objects**

To place database objects on the design window, select **DB Fields, Query**, and/or **Browse** objects from the Widgets section in the Palette. You are prompted to connect to a database, if the database connection is not configured with your current project.

**Placing pop-up menu or menu bar**

Use the **Popup Menu** or **Menu Bar** dialog to create a pop-up menu or menu bar for a selected object. See **Inserting Popup Menu or Menu Bar**.

**Placing OCX controls**

Use the OCX icon from the OCX section in the Palette to add OCX controls to the design window. See **Choose Control** dialog on page 755.

**Placing SmartObjects**

Use the **Open SmartObject** dialog to insert SmartObject instances on the design window. See **Inserting SmartObjects** on page 669.

**See also**

- **Choosing objects for the design window** on page 648
- **Palette options** on page 680

**Deleting objects from the design window**

To delete one or more objects from the design window, select them on the design window and press the **Delete** key. Alternatively, right-click on the design window and select **Delete** from the context menu.

When a control is deleted from the design window:

- If it is a container control, any child controls that it contains are also deleted.
- All code related to a deleted control is removed from the .w procedure file.
Note: If you want to restore a deleted object, select Undo option from the Edit menu.

For more information, see Choosing objects for the design window on page 648.

Defining properties and appearance

This section explains the techniques available for editing the appearance and behavior of the objects that have been placed on the design window.

Editing object properties

Each ABL object has its set of properties. You can modify the properties of a selected object using the Property Sheet dialog.

To edit the object properties:

1. Select an object on the design canvas, and click Property Sheet icon on the GUIDesigner toolbar. The Property Sheet dialog for the selected object appears.
2. Make the necessary changes to the object properties. See Property Sheet dialogs on page 702.
3. Click OK.

See also

Accessing property sheets on page 647
Property Sheet dialogs on page 702

Changing color of an object

Use the Choose Color dialog to change the background and foreground color of the design window or an object.

To change the color:

1. Do one of the following:

   • Click the Color icon from the GUI Designer toolbar.
   • Double-click an object on the design window to change the color of an object, and then click the Color icon on the Property Sheet dialog.

   The Choose Color dialog appears.
2. Select the foreground and background colors on the Choose Color dialog.
3. Click OK to save your changes.

Note: For more information, see Choose Color dialog on page 753

Changing font for an object

To edit font for an object, follow these steps:

1. In the design window, double-click on the object whose font you want to modify. The objects Property Sheet dialog appears.
2. Click Fonts icon. The Choose Font dialog appears.

3. From the available fonts list, select a font.

   **Note:** You can customize an available font, by selecting the font and clicking Edit. Font zero through seven are OpenEdge reserved fonts. You can customize only fonts eight through 255.

4. Click Save Font Settings to save custom font selections for future sessions.

5. Click OK to close the Choose Font dialog.

6. Choose OK to close the object's Property Sheet dialog.

   The object now displays the font you selected.

**See also**

Defining properties and appearance on page 650

Choose Font dialog on page 754

---

**Creating an Alternate Layout**

The Alternate Layout dialog is used to create and maintain layouts. Standard layout is an alternate layout with pre-defined run-time expressions.

To create an alternate layout:

1. Open a valid procedure file (.w) from the Project Explorer view.

2. Select Alternate Layout from the Layouts menu. The Alternate Layout dialog appears.

3. Do one of the following:
   - Select a standard layout from the Layout drop-down. Standard layout is an alternate layout with pre-defined run-time expressions.
   - If you want to create a new layout, click New. The New Layout Name dialog appears. Enter a name for the layout in the New Layout Name field.

4. Click OK. The Alternate Layout dialog reappears.

5. Do one of the following:
   - Select GUI to display the layout as graphical at design time.
   - Select Character to display the layout as character at design time.

6. Select the Activate Layout check box to activate the new alternate layout.

   **Note:** The Activate Layout option does not appear when you select layout type as Master Layout from the Layout drop-down in step-2.

7. Do one of the following:
• If you want to test a run-time expression, enter the run-time expression in the **Run-time Expression** field. Click **Check Syntax** to check the syntax of the expression.

• If you do not want to test a run-time expression, select the **Don't use Run-time Expression** check box.

**Note:** Any legal expression is acceptable, as long as it resolves to a LOGICAL value. However, if you use a character string within the expression, place the string in single quotes ("'") not in double quotes (" "). If you need to use embedded quotation marks, replace the quotation marks with the tilde (~) character. This convention helps the AppBuilder generate code that avoids Compiler ambiguities.

8. Enter a comment in the **Comment** field. This describes the alternate layout.

9. Click **OK** to save your changes.

The GUI Designer reloads your design window. The name of the alternate layout appears on the top of the design window.

**See also**

- Selecting standard or alternate layout on page 652
- Layout menu options on page 689

### Selecting standard or alternate layout

To select a standard or alternate layout:

1. Open a procedure file (.w) in an OpenEdge project from the Project Explorer view.
2. Select **Alternate Layout** from the Layouts menu. The **Alternate Layout** dialog appears.
3. Select one of the following:

   • **Standard layout** - If you select a standard layout, you cannot change the expression or name of the layout.
   • **Alternate layout** - If you select a user defined alternate layout, you can modify the expression and name.

4. Click **OK** to save the changes.

**See also**

- Creating an Alternate Layout on page 651
- Layout menu options on page 689

### Aligning objects

Use the Align submenu to align basic objects and SmartObject instances in the design window. Align function works only on selected objects or SmartObject instances that are parented to a common object.

To align objects:

1. In the design window, select an object or group of objects.
2. Select **Align** from the **Layout** menu. The **Align** submenu appears.
3. Select any of the following options:

| Colons | Aligns the label colons (:) of the selected basic object or SmartObject instance. |

**Note:** Any legal expression is acceptable, as long as it resolves to a LOGICAL value. However, if you use a character string within the expression, place the string in single quotes ("'") not in double quotes (" "). If you need to use embedded quotation marks, replace the quotation marks with the tilde (~) character. This convention helps the AppBuilder generate code that avoids Compiler ambiguities.
### Left Sides
Aligns the left sides of the selected basic object or SmartObject instance to the left edge of the leftmost basic object or instance.

### Vertical Centers
Aligns the vertical midpoints of all selected basic objects and SmartObject instances.

### Right Sides
Aligns the right sides of the selected basic object or SmartObject instance to the right edge of the rightmost basic object or instance.

### Top Edges
Aligns the selected object or SmartObject instances to the top edge of the topmost basic object or instance.

### Horizontal Centers
Aligns the horizontal midpoints of all selected objects and SmartObject instances.

### Bottom
Aligns the selected basic object or SmartObject instance to the bottom edge of the bottom-most basic object or instance.

### See also
- Working with objects on page 647
- Layout menu options on page 689

### Setting tab order
The Tab Editor dialog is used to change the tab order of an object in a selected frame. The Tab Editor lists the tab order of tappable objects which are child objects of the selected frame, and allows you to change the order.

You can change the tab order of objects within a child frame by selecting the child frame before opening the Tab Editor.

**Note:** The Tab Editor does not support dynamic objects (ABL widgets that you create with the CREATE Widget statement). You must use the appropriate ABL methods and attributes to program the tab order of dynamic widgets at run-time.

To change the tab order of objects:

1. Open Tab Editor by doing one of the following:
   - Select Tab Order from the Layout menu.
   - Click Tab Order icon in the object's Property Sheet dialog.
   - Right-click the object and select Tab Order from the context menu.
2. Select a tabbing option on the Tab Editor dialog on page 699.
3. Click OK to save the changes.
See also
- Working with objects on page 647
- Tab Editor dialog on page 699
- Layout menu options on page 689

Selecting translation attributes for an object

The **String Attributes** dialog helps you to set translation attributes for the selected object.

Translation attribute is specifying the translation format characteristics (justification and maximum length) for an object's label, title, help, or display format. The OpenEdge Translation Manager uses these settings when translating the character string into another language.

To set translation attributes for an object:

1. From the object's **Property Sheet** dialog, click the **Translation Attributes** icon. The **String Attributes** dialog appears.

2. Specify an attribute value for the object's characteristics such as Label.

   **Note:** The attribute must have one of L, R, T, or C and/or a U and/or one integer value. For example, R1 as the attribute value for Label.

3. Click **OK** to save your changes.

   **Note:** For more information, see **String Attributes dialog** on page 782

Resizing the design window

You can control or define the size of the design window in either of two ways:

- Click and drag the arrow handles located on the right and bottom edges of the design window.

- To set precise dimensions in pixels, enter the height and width values in the **Property Sheet** dialog. See **Property Sheet for Window** on page 702.

See also
- Opening ABL GUI procedure files on page 645
- Working with objects on page 647
Working with the code

Viewing and editing the source code

The OpenEdge ABL Editor is the default design editor for a valid procedure .w files. The valid .w files are the procedure files that are created using GUI Designer or standalone AppBuilder. This editor is used to edit the ABL code of such valid (.w) files.

Note: You cannot edit the auto-generated code of an AppBuilder file. The auto-generated code is read-only. See Read-only code sections of AppBuilder-generated code on page 635.

To display the source code editor while editing a design window in the ABL UI Designer, do one of the following:

- Select the View Source icon from the toolbar.
- Press F9.
- Right-click in the design window, and then select View Source from the context menu.

This displays the source code of the complete procedure file. If you want to view only the section of the code of an object, see Viewing and editing code of an object on page 648.

Conversely, to display the design window while editing the source code, do one of the following:

- Right-click in the design window and select View Design from the context menu.
- Press Shift+F9.

An asterisk * icon appears on the design view and the source view (if opened), when you make changes to either of the views. The changes are synchronized when you switch between the design canvas and the source view, and when you save the file. Thus, changes that you make with either editor are immediately reflected in the other.

Once you have opened the procedure file in either the source editor or design window, it remains open in that editor until you explicitly close it.

Note: When editing the code in the source view, make sure that you do not introduce errors in the UI code. It is advisable to use the design window for all changes to the UI, and edit only application logic in the source editor.

See also

- Code analysis in GUI Designer on page 633
- Coding conventions to generate AppBuilder code on page 634
- Editing ABL procedure files on page 645

Adding triggers

The Add Trigger dialog helps you to associate an event (also called as a trigger) with an object in the AppBuilder (.w) file. An event is an action associated with a trigger and on an object.

Note: The Add Trigger option is available only with the AppBuilder (.w) files.
To add trigger for an object:

1. Open the .w procedure file in the ABL Editor.
2. Open the Add Trigger dialog by doing one of the following:
   - From the Source menu, select Add Trigger.
   - From the right-click context menu, select Source > Add Trigger.

   The Add Trigger dialog appears.
3. From the Widget drop-down, select an object. This drop-down contains all the objects that are present in the current design window.
4. From the Event category drop-down, select an event category. For details on the event categories, see Add Trigger dialog on page 757.
5. If you select an event category other than Keyboard Event from the Event category drop-down, go to step 6 on page 656. If you select Keyboard Events as the event category, follow these steps:
   a) In the Key label field, enter a label for the keyboard event such as F1 or F2.
   b) The Key function displays the pre-defined function associated with the key label such as Help for F1 or GO for F2. If there is no pre-defined function associated with the Key label, then the same key label text is displayed in the Key function field.
   c) For the Generate code by option, select either Key Function or Key label.
   d) Go to step 7 on page 656.
6. From the Event drop-down, select an event. This drop-down contains all the events based on the object and event category you have selected.
7. Click Generate.

   The ABL Editor inserts the trigger code for the selected object in the (.w) procedure file.

See also
Add Trigger dialog on page 757

Renaming an internal procedure or function

The Rename option allows you to rename an internal procedure or function which is defined in an AppBuilder (.w) procedure file. You can rename only a user-defined function or procedure. This option is applicable only with an AppBuilder (.w) procedure file.

While renaming a function or procedure, you must follow these rules:

- Name must be alphabetic or alpha-numeric.
- Name must start with an alphabet (Name cannot start with numeric or special character).
- Name cannot include special characters.

Note: You cannot rename a function or procedure which is auto-generated by AppBuilder.

To rename an internal procedure or function:

1. Open an AppBuilder (.w) procedure file with the ABL Editor.
2. Select a function or procedure name that you want to rename, and then right-click on the selection.
3. Do one of the following:
   • From the right-click context menu, select **Refactor > Rename**.
   • From the **Source** menu, select **Refactor > Rename**.

   **Note:** The function or procedure name is highlighted with a message stating as: Enter a new name, and press **Enter** to rename.

4. Enter a name in the selection area.
5. Press **Enter**.

   All the references of the selected function or procedure are updated with the specified name, including the instances which are part of the read-only code sections.

**See also**
- Viewing and editing the source code on page 655
- Using the Find/Replace dialog on page 658

**Using the Outline view**

You can use the Outline view to easily navigate to a specific place in the code or widget in the design window. This view shows a tree view of the widgets or structural code elements of your AppBuilder procedure (.w) file that is currently open.

   **Note:** If the Outline view is not visible in the OpenEdge AppBuilder perspective, open it by selecting **Window > Show View > Outline**.

The Outline view displays elements depending on the currently open view of your procedure file, such as:

• When you work with source, this view lists all the structural elements of the procedure file such as Definitions, Main Block, Variables, Triggers, Procedures, Function Prototypes, and Function Implementations.

• When you work with UI designer, this view lists all the widgets present in the design window in a hierarchical tree structure. For example, Frame widget is shown as the parent node and all the widgets added to the Frame widget as its child nodes.

To navigate using the Outline view:

• Do one of the following:
  • Click on a node to take you to the corresponding location in the code or design window.
  • Double-click a widget to open its **Property Sheet** dialog. See preferences page to set to open **Property Sheet** dialog when you double-click an object.

**See also**
- Accessing the AppBuilder perspective on page 641
- Opening ABL GUI procedure files on page 645
- Outline view on page 813
Toggling to the Section Editor mode

The Section Editor mode enables you to view a block of code of an AppBuilder (.w) procedure file while navigating to specific places in the code in the Outline view, instead of displaying the entire code of the procedure file.

**Note:** The Outline view is not available with the design view. The Outline view lists all the structural elements of a procedure file only when the file is opened with the ABL Editor.

To toggle the Section Editor mode:

1. Open an AppBuilder procedure (.w) file in the ABL Editor.
2. In the Outline view, click the **Section Editor Toggle** icon.

**See also**

- Outline view on page 813
- Viewing and editing code of an object on page 648

Using the Find/Replace dialog

The **Find/Replace** dialog allows you to search and replace a string in the current (.w) procedure file. This option works in both the View Source and Section Editor modes.

When working in the Section Editor mode, the search is limited to only the code section that is open. The search does not run through the complete code of the procedure file.

**Note:** You can search for a string through both the read-only and editable code of an AppBuilder file, but the replace options are available only with the editable sections of the code.

To use the find and replace option:

1. Open an AppBuilder procedure (.w) file in the source view.
2. Open the **Find/Replace** dialog by doing one of the following:
   - From the main menu, click **Edit > Find/Replace**.
   - Press **CTRL+F**.

**Note:** The **Find/Replace** option is not available with the read-only or auto-generated code sections of the AppBuilder procedure file.

The **Find/Replace** dialog appears with a list of search and replaces options.

3. In the **Find** field, enter a search string. This is an instance of code or text you want to search in the procedure file.

**Note:** You can also select text in the procedure file, and then click **CTRL+F**. The **Find/Replace** dialog appears with the selected text in the **Find** field.

4. In the **Replace with** field, enter a replacement string. This is the code or text you want to use to overwrite one or all instances of the search string.
5. In the **Direction** section, select one of the following:
   - **Forward** - to search from the insertion point to the end of the file.
   - **Backward** - to search for the string from the insertion point toward the beginning of the file.

6. In the **Scope** section, select one of the following:
   - **All** - to search the entire/through all the code sections in a procedure file.
   - **Selected lines** - to search in a selected section of code in a procedure file.

7. In the **Options** section, select any of the following check boxes:
   - **Case sensitive** - to search only occurrences of the text that match the case of the string specified in the **Find** field.
   - **Wrap search** - to continue searching from the beginning to the end of the file, once it reaches the end (depending on the search direction).
   - **Whole word** - to look for the entire specified search string.
   - **Incremental** - to progressively search and filter through text in the active editor. As you type the search expression, it will incrementally jump to the previous exact match in the active editor.
   - **Regular expressions** - to search using wildcards or regular expressions. If you select this check box, you can specify whether the search will include wildcards or regular expressions, in the **Find** field.

8. Click **Find** to start searching for the string specified in the **Find** field. After an item is found, click one of the following buttons:
   - **Replace** - to replace only the current instance of search string.
   - **Replace/Find** - to replace the current occurrence of the search string and find the next occurrence.
   - **Replace All** - to replace all occurrences of the search string in the current (.w) procedure file.

9. Click **Close**.

**See also**
- Viewing and editing the source code on page 655
- Renaming an internal procedure or function on page 656
- Toggling to the Section Editor mode on page 658

**Selecting a display color for the read-only code**

The **Colors** page allows you to change the background color of the AppBuilder-generated read-only code sections in the ABL Editor.

To change the background color of the AppBuilder-generated read-only code:

1. Access the **Color** page by following these steps:
   a) From the main menu, select **Window > Preferences**. The **Preferences** window appears.
   b) Select **Progress OpenEdge > AppBuilder > Editor**. The **Editor** page appears.

2. Click the color icon beside the **Read-only code background color** option. The **Color** dialog appears.

3. Do one of the following:
   - Select from the list of available basic or custom colors.
• Click **Define Custom Colors** to define a custom color from the color palette.

4. Click **OK** to save your changes.

**See also**

- Read-only code sections of AppBuilder-generated code on page 635
- Editor preferences on page 678

### Working with objects Palette

#### Choosing the Palette docking position

By default, the Palette is docked at the right edge of the ABL UI Designer editing area. You can choose to dock the Palette at the right or left edge of the ABL UI Designer, as needed.

To specify a docking position for the Palette, right-click on the Palette title and select **Dock On > Left** or **Dock On > Right** from the context menu.

Alternatively, you can drag-and-drop the Palette on either the right or left edge of the ABL UI Designer editing area.

**Note:** The specified docking position will be persistent the next time you open Progress Developer Studio for OpenEdge.

**See also**

- Working with objects Palette on page 660
- Object Palette on page 680

#### Changing the Palette settings

The Palette Settings dialog enables you to modify the display of the objects in the Palette.

To change the Palette settings:

1. Right-click on the Palette and select **Settings** from the context menu. The **Palette Settings** dialog appears.
2. Click **Change** to open the **Font** dialog. This dialog allows you to change the font of the controls available on the Palette.

**Note:** If you want to restore the default font settings for the Palette controls, select **Restore Defaults**.

3. Select the layout type for the **Palette** options from the **Layout** section.
4. Select **Use Large Icons** check box to increase the size of the controls available on the Palette.
5. Select the drawer type from the Drawer options section. The drawer options allow you to set the layout of the sections (Widgets, OCX, and SmartObjects) in the Palette. The following options are available:
   • Always close when opening another drawer
   • Close automatically when there is not enough room
Never close

6. Click OK.

See also
Working with objects Palette on page 660
Choosing Palette layout on page 661
Object Palette on page 680

Choosing Palette layout
You can change the layout of the Object Palette by specifying how the widgets must appear in the Palette.
To choose a layout:
1. Right-click on the Object Palette.
2. From the right-click context menu, select Layout. The following layout options are available:
   - Columns - select to display the objects in a column format.
   - List - select to display the objects in a list format.
   - Icons Only - select to display the objects as only icons without the names of the objects.
   - Details - select to display the objects in bold.

Note: Alternatively, you can change the layout of the Object Palette using Palette Settings dialog. See Changing the Palette settings on page 660.

See also
Working with objects Palette on page 660
Object Palette on page 680

Customizing the Object Palette
In addition to providing a variety of display options for the Object Palette, the AppBuilder allows you to customize the Palette contents using custom object files.

The custom object file is an external text file in which you define custom objects using a specific syntax. Custom object files have .cst extension in their filenames. OpenEdge includes default custom object files for you to use such as progress.cst, smart.cst, activex.cst, and shared.cst. These files reside at DLC/src/template.

However, you can create your own custom object files, and add them to the Object Palette. For example, if you create new SmartObject templates, you can add them to the Object Palette. You can also change certain default settings for the standard SmartObject templates.

The Use Custom dialog allows you to add to or modify the files that determine the Object Palette's behavior and appearance.

To customize the Object Palette:
1. Right-click on the Object Palette, and then select Use Custom from the context menu. The Use Custom dialog appears which allows you to perform the following:
a) Click Add to add a custom object file.
b) Click Modify to modify an existing custom object file.
c) Click Remove to delete an existing custom object file from the list.
d) Click Move Up or Move Down to change the order of a selected custom object file in the list.

2. Click OK to save your changes.

Note: The Object's Palette behavior and appearance changes based on the modifications you have done.

See also
Adding custom object files on page 672
Modifying custom object files on page 672
Use Custom dialog on page 682

Working with Progress Dynamics

Starting AppBuilder with Dynamics enabled

You can use AppBuilder, running embedded in Progress Developer Studio for OpenEdge, to develop Progress Dynamics applications. Progress Dynamics applications are built from objects that reside in the Progress Dynamics Repository. The Progress Dynamics Repository consists of an OpenEdge database named icfdb whose schema is defined to store information about OpenEdge applications.

The prerequisites for running AppBuilder as a Progress Dynamics application development tool embedded in Progress Developer Studio for OpenEdge are:

- Select the Progress Dynamics option during installation of Progress Developer Studio for OpenEdge - Progress Dynamics is an optional component.
- Configure Progress Dynamics - Usually you run the Dynamics Configuration Utility, which is offered as an option immediately after installation. For more information see the OpenEdge Getting Started: Installation and Configuration manual. You can find OpenEdge manuals in the Product Documentation section of the Progress Software Developer's Network Web site.
- Start the Dynamics Repository - You can start the Repository from the Windows Start menu, typically: Start > Programs > OpenEdge > Progress Dynamics > Start Dynamics DB Servers.

To run AppBuilder as a Progress Dynamics application development tool embedded in Progress Developer Studio for OpenEdge:

1. Select a project name in the Project Explorer view of Progress Developer Studio for OpenEdge.
2. Select Properties from the context menu.
3. Select OpenEdge > Propath to display the Propath Properties page.
4. If Standard OpenEdge Dynamics Paths is not listed in the tree view, add it by clicking Add Standard Paths.
5. Select AppBuilder to display the AppBuilder properties page.
6. Choose Enable Progress Dynamics.
7. Enter the path to your icfconfig.xml file in the Configuration File field.
8. Choose a development session type from the Session Type box.

9. Click Apply and close the Properties page.

10. Open the OpenEdge AppBuilder perspective: Window > Open Perspective > OpenEdge AppBuilder.

11. Click the AppBuilder button in the toolbar. An Application Login dialog appears.

12. Enter a login name (the default is admin) and a password (left empty by default).

AppBuilder opens in the AppBuilder Designer view. You can access the Progress Dynamics administration and development tools from the AppBuilder Tools menu. You can find OpenEdge manuals, presentations, and Web papers about Progress Dynamics in the Product Documentation section of the Progress Software Developer's Network Web site.

**Starting Progress Dynamics session**

The Application Login dialog allows you to start a Progress Dynamics session in Progress Developer Studio for OpenEdge. This will enable to develop Progress Dynamics application in Progress Developer Studio for OpenEdge.

You can also use this dialog to change your password. The login program provides for application security and language control.

To start a Progress Dynamics session:

1. From the File menu, click Open Object. The Application Login dialog appears.

   **Note:**
   The Open Object option is available only for Dynamics project.

2. In the Login Name field, type admin. This is the default login name. The other fields on the Application Login dialog are optional. Leave the default values.

   **Note:** The Password button is disabled at the first login. This displays the Change User Password dialog which allows you to change the password, which you can use on subsequent logins.

3. Click OK.

   This starts the Progress Dynamics session in Progress Developer Studio for OpenEdge. For more information, see Setting Progress Dynamics projects on page 663

**Setting Progress Dynamics projects**

You must set up a Dynamics project with Progress Dynamics to develop Progress Dynamics applications in Progress Developer Studio for OpenEdge. The Progress Dynamics Repository provides dynamic objects which you can use to develop Progress Dynamics applications in OpenEdge.
The Progress Dynamics Repository stores data for a wide variety of purposes. Most application objects are represented as data stored in the Repository, so you do not need to create, compile, deploy, or maintain the source code for them. The framework includes programs that render these objects at run time from the data in the Repository.

Before you set up your dynamics project to work with Progress Dynamics, you must install and configure the OpenEdge Studio which includes Progress Dynamics. See *OpenEdge Getting Started: Installation and Configuration* for complete instructions. You can access this and other Progress Dynamics manuals in the *OpenEdge Product Documentation* section of the Progress Communities web site.

To set up a Progress Dynamics project:

1. Create a Dynamics project in OpenEdge. See Create a Dynamics project.
2. Enable Dynamics for the selected dynamics project. See AppBuilder properties page.
3. Establish the Repository database (ICFDB.db) connection to the Dynamics project. You must connect to ICFDB.db to access the Progress Dynamics tools in the Progress Developer Studio for OpenEdge. See Adding a new database connection to a projectAdding a new database connection to a project.
4. After you set up a Progress Dynamics project, start Dynamics DB Servers to start working with the Progress Dynamics objects. On the Windows, Start > All > Programs menu, select Progress > OpenEdge > Progress Dynamics > Start Dynamics DB Servers.

The Progress Dynamics DB Server is now started. You can get started working with the dynamics objects.

**See also**

- About Progress Dynamics on page 640
- Opening objects from the Repository on page 664
- Progress Dynamics menu options on page 690
- Progress Dynamic dialogs on page 808
- Dynamic objects on page 686

**Opening objects from the Repository**

The *Open Object* dialog allows you to browse for objects stored in the Repository and select an object for viewing or editing in the design window.

**Note:** The *Open Dialog* menu option is available only with the Dynamic projects. See Setting Progress Dynamics project.

To open a dynamics object:

1. In the Project Explorer view, select a dynamics project.
2. From the *File* menu, select *Open Object*. The *Open Object* dialog appears.
3. The Object browser displays the objects stored in the Repository for the selected product module or type, or that match the filename filter value.

**Note:** You can change how the browse displays information in each column by clicking on the column name. For example, if you click on the Object filename column, all columns are sorted and displayed by the object filename in alphabetical order. If you click on Object type code, all columns are sorted and displayed by object type code in alphabetical order. You can use any column to sort and display the information in the browse.
4. You can specify any of the following filter options to narrow down your search:

- **Object filename** field – Enter a full or partial object file name to automatically filter the objects listed in the Open Object dialog. As you enter each character, the list of objects in the browser narrows.
- **Module** drop-down – Select a product module to display objects created in that product module.
- **Type** drop-down – Select an object type to display objects of that type.
- **Object filename** drop-down – Displays the most recently opened object list. Select any recently opened dynamic object.

5. Select an object from the **Object browser** list, and click **Open**.

6. The selected dynamic object opens for editing in the GUI Designer design canvas. Additional options which are available on the right-click context menu of a dynamic object in the Object browser are:

- **Open** – Opens the selected object for editing in the GUI Designer design canvas.
- **Remove from Repository** – Displays an alert box to confirm the removal of the selected dynamic object from the Repository.
- **Properties** – Displays the Object Properties dialog which allows you to view properties for the selected object.

**See also**

About Progress Dynamics on page 640
Setting Progress Dynamics projects on page 663
Open Object dialog on page 808

**Editing and saving dynamic objects**

You can modify a dynamic object and save that dynamic object with the changes back in the Repository.

**Note:** The **View Source** option is disabled for dynamic objects. Therefore, you can modify a dynamic object using only the design view.

To edit and save a dynamic object:

1. Open a dynamic object. See Opening objects from the Repository on page 664. The selected dynamic object opens in the design view.
2. In the design view, modify the dynamic object as required. An asterisk * icon appears on the design window when you make any change to the dynamics object.
3. Press CTRL+S to save your changes.

You can re-open the dynamic object to view your changes.

**See also**

Opening objects from the Repository on page 664
Registering objects in Repository on page 668

**Saving dynamic objects as static**

The **Save As Static Object** dialog enables you to save a dynamic object as a static object and register that object to the Repository.
You can save the following dynamic objects as static: Dynamic SmartDataViewers and Dynamic SmartDataObjects.

**Note:** The **Save Dynamic Object As Static** action in the **File** menu is available only with the Dynamics projects, and when you have any of the above dynamic objects open in the design window. See **Setting Progress Dynamics projects**.

To save a dynamic object as static:

1. Open one of the above-mentioned dynamic objects using the **Open Object dialog** on page 808.
2. From the **File** menu, select **Save Dynamic Object As Static**. The **Save As Static Object** dialog appears.
3. In the **File Name** field, enter a name for the static object. By default, the name of the selected dynamic object appears in this field. You can enter a different name.

    **Note:** The name you have entered must be a unique object name in the Repository.

4. The **Save in** field displays the default path where the static object is saved as dynamic. Click the **Browse** icon to select a different path.
5. Select the **Register object** check box, if you want to register the dynamic object to the Repository.

    **Note:** The following fields are enabled only when you select the **Register object** check box:

    a) The **Object name** displays the object name. This is a read-only field.
    b) In the **Description** field, enter a description of the object.
    c) The **Root Directory** field displays the default root directory of the object. Click **Browse** to select a different root directory.
    d) From the **Product Module** drop-down, select a product module in which you want to save the static object as dynamic.
    e) The **Full path name** field displays the complete location where the static object is saved as dynamic. This is read-only field.
    f) Select the **Deploy to Web** check box to deploy the object to web (for Web and client/server deployment types).
    g) Select the **Deploy to Client** check box to deploy the object to the client machines (for WebClient and client/server deployment types).
    h) Select the **Deploy to server** check box to deploy the object to remote servers (for client/server, AppServer, and Web deployment types).
    i) Select the **Design object** check box to deploy the object only for design (development) purpose.

6. Click **Save** to save your changes.

    The selected dynamic object is saved as a static object to the Repository.

**See also**

- Saving static objects as dynamic on page 667
- **Save As Static Object dialog** on page 810
- **Progress Dynamics menu options** on page 690
Saving static objects as dynamic

The **Save As Dynamic Object** dialog enables you to save a static object as a dynamic object.

You can save the following static objects as dynamic objects: Static SmartDataViewers (StaticSDV), Static SmartDataBrowsers (StaticSDB), and Static SmartDataObjects (SDO).

**Note:** The **Save Static Object As Dynamic** action in the **File** menu is available only with the Dynamics projects, and when you have any of the above static objects open in the design window. See [Setting Progress Dynamics projects](#).

To save a static object as dynamic:

1. Open one of the above-mentioned static objects using the **Open Object** dialog on page 808.
2. From the **File** menu, select **Save Static Object As Dynamic**. The **Save As Dynamic Object** dialog appears.
3. The **Dynamic Object Type** drop-down displays the default object type of the selected object.
4. In the **Object Name** field, enter a name for the dynamic object. By default, the name of the selected static object appears in this field. You can enter a different name.

   **Note:** The name you have entered must be a unique object name in the Repository.

5. In the **Description** field, enter a description of the object.
6. From the **Product Module** drop-down, select a product module in which you want to save the object.
7. The **Relative Directory** drop-down displays the PROPATH-relative directory of the selected product module.
8. Select the **Create Custom Super Procedure** check box to create a custom super procedure for the object.

   **Note:** The default custom super procedure name appears in the **File Name** field.

9. The following fields are enabled only when you select the **Create Custom Super Procedure** check box:
   a) The **Root Directory** field displays the default root directory of the custom super procedure. Click **Browse** to select a different root directory.
   b) The **File Name** field displays the default custom super procedure name of the object. You can enter a different name.
   c) The **Type** field displays the type of custom super procedure, as Procedure.
   d) From the **Product Module** drop-down, select a product module in which you want to save the custom super procedure.
   e) The **Relative Directory** drop-down displays the PROPATH-relative directory of the selected product module.
   f) The **Full Path Name** field displays the complete location where the custom super procedure is saved. This is read-only field.

10. Click **OK** to save your changes.

    The static object you have selected is saved as a dynamic object in the Repository.
Registering objects in Repository

The Register in Repository dialog allows you to register GUI Designer created static objects to the Repository.

**Note:** The Register in Repository menu is enabled only for static objects which are not already stored in Repository, and with a Dynamics project. This menu action is not enabled for dynamic objects.

To register a static object to the Repository:

1. Open a static object from your Dynamics project:
   - In the Project Explorer view, right-click on a static object.
   - Select Open With > ABL UI Designer from the right-click context menu.
     The selected static object opens in the design window.
2. From the File menu, select Register in Repository. The Register in Repository dialog appears.
3. The File field displays the name of the selected static object. This is a read-only field.
4. From the Product module drop-down, select a product module for the object you are registering.
   **Note:** Product module is a group that defines any useful type of organization for your Progress Dynamics application. You set up Products and Product Modules using the Application > Product Control > Dynamics Administration menu option in the window.
5. From the Object type drop-down, select an object type.
6. The Relative path field displays the relative path of the directory in which the selected static object was saved.
7. Select the Deploy to Web check box to deploy the object to web (for Web and client/server deployment types).
8. Select the Deploy to Client check box to deploy the object to the client machines (for WebClient and client/server deployment types).
9. Select the Deploy to server check box to deploy the object to remote servers (for client/server, AppServer, and Web deployment types).
10. Select the Design object check box to deploy the object for design (development) purposes only.
11. Click OK.
   **Note:** The static object is registered in the Repository.

You can open the registered objects using the Open Object dialog. See Opening objects from the Repository on page 664.
Working with PRO*Tools

Adding user-defined utilities

In the standalone AppBuilder, you can add any user-defined executable ABL program using protools.dat. The protools.dat file contains all the programs information to display the PRO*Tools palette in the standalone AppBuilder. Any modifications to the protools.dat file will not be available with AppBuilder in Progress Developer Studio for OpenEdge.

Alternatively, you can add any user-defined executable ABL program as a menu/toolbar action in Progress Developer Studio for OpenEdge, using OpenEdge Customization Editor. You can access OpenEdge Customization Editor by selecting the menu: OpenEdge > Tools > Customization Editor.

For more information on adding a menu/toolbar item using Customization Editor, see Using the Customization Editor and Menu and toolbar customization.

Note: Any new PRO*Tools utility added in Progress Developer Studio for OpenEdge will not be available with standalone AppBuilder.

For more information, see PRO*Tools utilities on page 695

Showing or hiding toolbar buttons and menu items

You can show or hide the AppBuilder, Progress Dynamics, and PRO*Tools actions which are visible on the toolbar and from the menu in your project perspective, using the Customize Perspective dialog.

You can access the Customize Perspective dialog from the menu by clicking Windows > Customize Perspective. Use the Tool Bar Visibility and Menu Visibility tabs in the dialog to hide or show the toolbar buttons and menu items visible in your project perspective.

See also
Working with PRO*Tools on page 669
Toolbar and menu options on page 688
PRO*Tools utilities on page 695

Inserting SmartObjects

The Open SmartObject dialog allows you to select SmartObject master on page 638 to use to create a SmartObject instance on page 638 to insert on the design window.

To build a SmartObject, using the Open SmartObject dialog, create a SmartObject instance, insert the SmartObject instance into a SmartContainer, and link the SmartObject to other SmartObjects, as needed.

To create SmartDataBrowsers, SmartDataObjects, and SmartDataViewers:
1. From the File menu, select New > Other.
   The New wizard appears.

2. From the list, select ABL UI Design as the file type in the Progress OpenEdgeAppBuilder node and click Next.
   The New ABL UI Design wizard appears.

3. From the ObjectType list, select the object you want to create.

4. Click Finish and follow the wizard.
   The selected Smart object is created and opens with the design window.

**See also**
- Creating ABL GUI procedure files on page 644
- SmartObjects on page 684
- Open SmartObject dialog on page 788

### Running ABL procedure files

Progress Developer Studio for OpenEdge provides support for running and debugging ABL procedure (.w) files using launch configurations. A launch configuration determines the characteristics of the ABL Virtual Machine (AVM) session in which the program runs. See Running and Debugging ABL Programs.

To run an ABL procedure (.w) file:

1. Open a procedure (.w) file that you want run with ABL UI Designer. See Opening ABL GUI procedure files on page 645.

2. Do one the following:
   
   • Run the currently selected procedure file by clicking Run on the toolbar.
   • Launch the Progress Developer Studio for OpenEdge Debugger and debug the current procedure file by clicking Run on the toolbar.

**See also**
- Creating ABL GUI procedure files on page 644

### Inserting Popup Menu or Menu Bar

The Popup Menu or Menu Bar Property Sheet dialogs allow you to create pop-up menu or menu bar for a selected object.

To create a pop-up menu or menu bar:

1. Double-click an object for which you want to create a menu. The Property Sheet dialog for the selected object appears.

2. Choose the Popup Menu icon or the Menu Bar icon on the object Property Sheet dialog. The menu Property Sheet dialog for that object appears.

3. In the Title field, enter a title for the menu.
4. In the Label field, enter a name for the menu.

   Note: The other fields are optional. You can edit the attributes on this dialog, if required.

5. Click OK.

See also
Creating ABL GUI procedure files on page 644
Types of objects on page 683
Property Sheet dialogs on page 702

Working with method library reference

Adding method library reference

Use the Add Method Library Reference dialog to add a method library to the procedure.

To add a method library to the procedure:

1. Click Add on the Method Libraries dialog.

   The Add Method Library Reference dialog appears.

2. Do one of the following:

   • Enter the name of a method library in the Include Reference field.
   • Click Files to browse and select a method library.

3. Click OK to save the changes.

See also
About Method Library on page 639
Modifying method library reference on page 671
Method Libraries dialog on page 780

Modifying method library reference

Use the Modify Method Library Reference dialog to modify a method library to the procedure.

To modify a method library added to the procedure:

1. On the Method Libraries dialog, select a method library from the Method Library Include References list.
2. Click Modify. The Modify Method Library Reference dialog appears.
3. Do one of the following:

   • Enter the name of a method library in the Include Reference field.
   • Click Files to browse and select a method library.

4. Click OK to save the changes.
Working with custom object files

Adding custom object files

The Add Custom Object File dialog allows you to add a custom object file to the list of custom object files used by the GUI Designer.

The custom object files normally have a .cst extension in their filenames.

To add a custom object file:

1. Click Add on the Use Custom dialog.
   - The Add Custom Object File dialog appears.
2. Do one of the following:
   - Enter a file name of the custom object file in the File Name field.
   - Click Files to open the Find File dialog, and then browse directories to select a custom object file.
3. Click OK to save your changes.

See also
Adding custom object files on page 672
Use Custom dialog on page 682

Modifying custom object files

The Modify Custom Object File dialog allows you to modify the file names of the custom object files available in the Custom Object Files list.

To modify a custom object file:

1. Select a custom object file from the Custom Object Files list.
2. Click Modify. The Modify Custom Object File dialog appears.
3. Do one of the following:
   - Edit the custom object file name in the File Name field.
   - Click Files to select a different custom object file, by browsing the directories.
4. Click OK to save your changes.

See also
Adding custom object files on page 672
Modifying custom object files

The **Modify Custom Object File** dialog allows you to modify the file names of the custom object files available in the **Custom Object Files** list.

To modify a custom object file:

1. Select a custom object file from the **Custom Object Files** list.
2. Click **Modify**. The **Modify Custom Object File** dialog appears.
3. Do one of the following:
   - Edit the custom object file name in the **File Name** field.
   - Click **Files** to select a different custom object file, by browsing the directories.
4. Click **OK** to save your changes.

See also

- Adding custom object files on page 672
- Use Custom dialog on page 682

Viewing ABL messages

The ABL Messages view enables you to look up information about the ABL error messages in Progress Developer Studio for OpenEdge.

**Note:** The ABL Messages option is available with only the OpenEdge projects.

To view an OpenEdge message:

1. From the Help menu, click **ABL Messages**. The ABL Messages window appears as a view in the AppBuilder perspective.
2. In the **Enter message number** field, enter an error number.
3. Click **Go to locate** and display the message text for the error number entered above.

   The Search result section displays the synopsis, category, and description for the error message ID. For example, when you type the error number as 104 in the **Enter message number** field and click **Go**, the **Search result** section displays the message description as: Invalid characters were replaced with blanks. (104). The PROGRESS editor does not allow non-printing control characters or tabs, and replaces them with blanks. For more information, see **ABL Messages view** on page 816.
Opening AppBuilder Properties view

The AppBuilder Properties view displays the attributes of one or more selected widgets. When more than one object is selected, this window makes available the cross-section of the attributes of the selected objects. This view allows you to do the following:

- Modify an attribute of one or more selected widgets.
- Change widget type of an existing widget such as change Fill In widget to Editor.

To open the AppBuilder Properties view:

- From the main menu, choose Window > Show View > AppBuilder Properties.

Enabling AppBuilder shortcut keys

Most of the shortcut keys in Progress Developer Studio for OpenEdge (Developer Studio) differ from AppBuilder shortcut keys. If you want to use standalone AppBuilder shortcut keys in the integrated environment of AppBuilder (in Developer Studio), you can apply many of the AppBuilder key bindings by setting a single workspace preference.

For more information on general behavior of keys, see the Keys topic in the Workbench User Guide.

To enable AppBuilder shortcut keys:

1. Open the Preferences window by choosing Window > Preferences from the main menu bar.
2. Expand the General node in the tree view.
3. Select Keys.
4. Select AppBuilder in the Scheme box. For

That the Keys preference page lists the commands, the key bindings, and the context where the key binding is in effect. It also allows you to customize individual key bindings.
**Note:** Not all AppBuilder shortcut keys are enabled. For example, F5 for previewing code is not implemented because it is unnecessary. Code is always displayed in the ABL editor when you open a .w file with AppBuilder embedded in Progress Developer Studio for OpenEdge. You can check the bindings for a particular key, key sequence or command by typing in the type filter text box. For example, typing F5 will show all the key sequences that use the F5 key.

5. Click **OK**.

6. If you are running Progress Developer Studio for OpenEdge in an external Eclipse workbench (an Eclipse workbench other than the one that ships with Progress Developer Studio for OpenEdge), some configuration is required before you can enable AppBuilder shortcut keys:
   a) In Windows Explorer, navigate to DeveloperStudio-install-dir/eclipse/plugins (the default folder for Progress Developer Studio plugins).
   b) Copy the folder com.openedge.keybinding.hook and the file com.openedge.keybinding.hook_1.0.0.jar to the ../eclipse/plugins folder of the external Eclipse workbench.
   c) Open ../eclipse/configuration/config.ini in the additional Eclipse installation with a text editor.
   d) Set osgi.framework.extensions=com.openedge.keybinding.hook.

**Note:** When you restart the external Eclipse workbench, you can enable AppBuilder shortcut keys.

---

**Specifying DB-Required code blocks**

The DB-Required option is used in data objects that need to support both a connected and disconnected mode (that is an AppServer client). Specifying DB-Required wraps the code block in a conditional preprocessor directive allowing it to be excluded from the compilation conditionally.

Data aware objects (SDO, SBO, and Data Logic Procedures) are automatically created with a client version and full (server) version, where the client version defines the preprocessor directive that excludes all DB-Required code blocks.

The DB-Required setting must be specified when a code block contains database field or table references that require the data object to be connected to appropriate databases at run-time in order for the code block to execute. This will ensure that this code is excluded from the client proxy r-code, All code blocks that are not DB-Required, are compiled into both r-code files.

**Note:** In Progress Developer Studio for OpenEdge new internal procedures and functions are not DB-Required by default. This is different from the standalone AppBuilder, which default to add the DB-Required option to new procedures and functions.

To change a procedure or function's DB-Required setting:

1. From the **Outline** view, select an internal function or procedure.
2. Toggle the DB-Required option in the view.

This adds or removes the DB-Required block generated for the selected function or procedure, depending on the toggle option you have selected on the context menu.

**See also**

Outline view context menu options on page 814
Editing a Freeform Query

When you create Freeform queries for Query Builder widget using the Freeform Query option, the nodes in the Outline view are represented by icon to differentiate it from the other nodes.

**Note:** You can edit the Freeform query for the workspace files only.

You can edit a freeform query as follows:

1. Open the .w file in the source view.
2. Right-click the Query widget in the Outline view and select **Edit Freeform Query**.
   
   The **Edit Freeform Query** dialog box opens.
3. Edit as required and click OK.

**Caution:** If you are not connected to the database, content assist and color coding will not work as expected.

Reference

AppBuilder perspective components

The major components of the OpenEdge AppBuilder perspective are the Project Explorer View, Design Window, Object Palette, Progress Dynamics actions, PRO*Tools utilities, Outline View, ABL Messages View, and ABL Cue Cards View.

**Project Explorer View**

The Project Explorer view provides a hierarchical view of the projects and resources in your workspace. From here, you can add new AppBuilder procedure (.w) files to your OpenEdge project or open them for editing.

**Design Window**

Design window serves as a working area where you insert basic objects, SmartObject, and OCX (ActiveX) controls while building your application user interfaces. It provides an accurate visualization of how the window will appear at run time. You can use the mouse to size objects by dragging their border, and to position controls on the form.

**Object Palette**

The GUI Designer has a new dockable Palette which contains controls with which you build the UI. Use the Palette to insert basic objects, SmartObjects, and OCX (ActiveX) on the design window.

The Palette makes available all the component objects supplied with GUI Designer. Use the Palette to insert basic objects, SmartObjects, and OCX (ActiveX) to the design canvas.

**Note:** Only the major organizers (windows, dialogs, SmartFrame) are not on the Palette. Create them from the list in the ABL UI Design dialog (File > New > ABL UI Design).
Progress Dynamics framework
The Progress Dynamics framework is a comprehensive, repository-based environment for Progress Developer Studio for OpenEdge users who are building new applications, or who need to migrate existing applications to take advantage of new OpenEdge technologies.

PRO*Tools Utilities
The PRO*Tools are a set of utility programs that you might need while developing your applications. The PRO*Tools utilities are available from the OpenEdge menu and as toolbar buttons in your OpenEdge AppBuilder perspective.

Outline View
A list of the structural elements of the AppBuilder procedure (.w) file that is currently open in the ABL Editor or ABL UI Designer. You can use the Outline view as an alternative means of selecting, copying, deleting, and renaming widgets.

ABL Messages View
The ABL Message view enables you to look up information about the OpenEdge error messages in Progress Developer Studio for OpenEdge. This view is available with only the OpenEdge projects.

ABL Cue Cards View
The ABL Cue Card view displays help information for a specific SmartObject type. When you create a new SmartObject master, the Cue Card associated with the SmartObject type is also opened. The Cue Card provides a basic definition of the SmartObject type, along with information about how to create and use the object.

See also
Introducing the ABL GUI Designer on page 631
OpenEdge AppBuilder perspective on page 632
Accessing the AppBuilder perspective on page 641

Preferences page

AppBuilder preferences
The AppBuilder preferences page allows you to set general options of AppBuilder in Progress Developer Studio for OpenEdge.

This page includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>When double-click on object</td>
<td>Specify to open the object Properties Sheet dialog or the edit code editor, when you double-click an object.</td>
</tr>
</tbody>
</table>

Note: On double-clicking a SmartObject, the Properties Sheet dialog opens even though the Edit code editor option is selected.
**Option** | **Description**
--- | ---
Default function data type | Specify a data type to a new function. The default value is Character.

Display these Advisors | Control the display of Progress Advisors. The following options are available:
- **Currently activated** - display only those Progress Advisors for which the Don't tell me this again check box is not selected.
- **All** - display all Progress Advisors.
- **None** - do not display any Progress Advisors. This is the default selected option.

Restore Defaults | Reset the preferences to the default values.

Apply | Apply the changes you made to the current workspace configuration.

---

**See also**
- Setting AppBuilder preferences on page 642
- Preferences page on page 677

**Editor preferences**

The **AppBuilder Editor preferences** page provides options to control the display behavior of AppBuilder-generated code in the ABL Editor.

**Note:** You can make the sections of the AppBuilder-generated code read-only in the ABL Editor. Thereby, you can ensure that the AppBuilder-generated code is not overwritten. Editing the AppBuilder-generated code might make the AppBuilder procedure (.w) file corrupt, and display errors when trying to open with the ABL UI Designer.

This page includes the following options:

| Option | Description |
--- | ---
Make AppBuilder generated code read-only | Select this check-box to mark the AppBuilder-generated code as read-only in the ABL Editor.

**Note:** By default, the **Make AppBuilder-generated code read-only** check-box is selected.

Read-only code background color | Display the **Color** dialog, which allows you to select a display color for the read-only code blocks in the ABL Editor. The default display color is gray.

**Note:** This option is enabled when the **Make AppBuilder generated code read-only** check-box is selected.
Grid Units preferences

The Grid Units preferences page allows you to set grid display options of the designer window. Grid is a framework of lines used to align objects within frames.

This page includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display grid</td>
<td>Allow you to show grid on the design window.</td>
</tr>
<tr>
<td>Snap to grid</td>
<td>Allow you to move and resize objects only from grid point to grid point.</td>
</tr>
<tr>
<td>Grid layout unit size (Graphical Windows only)</td>
<td>Specify the width and height of the grid unit.</td>
</tr>
<tr>
<td>Grid units between major lines</td>
<td>Specify the number of grid units to set the space between horizontal and vertical grid lines. This value must be an integer.</td>
</tr>
<tr>
<td>Restore Defaults</td>
<td>Reset the preferences to the default values.</td>
</tr>
<tr>
<td>Apply</td>
<td>Apply the changes you made to the current workspace configuration.</td>
</tr>
</tbody>
</table>

**Grid layout unit size (Graphical Windows only)**

**Note:** Units of measure (characters or pixels) is specified by the Layout Units option.

Grid layout unit size (Graphical Windows only) is a preference that allows you to specify the width and height of the grid unit. This is particularly useful when working with graphical user interfaces, as it helps in maintaining consistency and alignment of objects within frames.

See also

Read-only code sections of AppBuilder-generated code on page 635
Setting AppBuilder preferences on page 642
Preferences page on page 677

Widget ID preferences

The Widget ID preferences page allows you to set Widget ID options.
The options on this page control two different methods for assigning widget-ids. The Static widgets widget-id assignment group affects how the AppBuilder assigns widget-ids to non-Smart, static widgets. The Runtime widget-id group affects only how the Runtime WIDGET-ID Assignment PRO*Tool works.

This page includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static widget ID assignment</td>
<td>This group affects how the GUI Designer assigns widget IDs to non-Smart and static widgets.</td>
</tr>
<tr>
<td></td>
<td>The following options are available in this group:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Automatically assign widget IDs</strong> - The GUI Designer assigns WIDGET-ID attribute values by default when static widgets are created.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Starting widget ID for frames</strong> - Defines the value for the first frame in a container when the GUI Designer assigns the WIDGET-ID attribute.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Frame widget ID increment</strong> - Defines the amount by which the GUI Designer increases the value for each additional frame in a container.</td>
</tr>
<tr>
<td>Runtime widget ID</td>
<td>This group affects only how the Runtime WIDGET-ID Assignment PRO*Tool works.</td>
</tr>
<tr>
<td></td>
<td>The following options are available in this group:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Save widget ID file name</strong> - Assigns a value to the widgetIDFileName property when you first save a container. This setting has no effect after the first time you save the container. The GUI Designer stores the value in the static code or in the Progress Dynamics Repository.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Default</strong> - Defines the widgetIDFileName property as the default value, container_filename.xml. For example, the default XML file for the window wCust is wCust.xml.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Custom XML file</strong> - Displays an input field in which you can specify an alternate name for the XML file.</td>
</tr>
<tr>
<td>Restore Defaults</td>
<td>Reset the preferences to the default values.</td>
</tr>
<tr>
<td>Apply</td>
<td>Apply the changes you made to the current workspace configuration.</td>
</tr>
</tbody>
</table>

**See also**
- Setting AppBuilder preferences on page 642
- Preferences page on page 677

**Object Palette**

**Palette options**

Use the object Palette to insert basic objects, ActiveX objects, and SmartObject instances. You can also create SmartObject masters.
You can resize the Object Palette with your mouse, and you can also add custom basic objects and SmartObjects to the Palette.

**Note:** The object's Palette is available only with procedure .w file open in the ABL UI Designer window.

The object Palette includes the following controls:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hide Palette</td>
<td>Collapses the Palette pane.</td>
</tr>
<tr>
<td>Show Palette</td>
<td>Expands the Palette pane.</td>
</tr>
<tr>
<td>Pointer</td>
<td>Enables you to select an object on the design canvas and perform actions like resizing or double-clicking an object. When you make a selection on the Palette, the insertion mode is activated and displays the pointer as an icon representing the basic object or SmartObject type being selected. In insertion mode, the mouse pointer can only be used to draw and insert basic objects, SmartObject instances, and OCXs controls on the design canvas. In insertion mode, when the pointer is located on the design canvas, the GUI Designer displays the pointer as an icon corresponding to the object being inserted. You can lock an icon in insertion mode by double-clicking on an object icon in the Palette.</td>
</tr>
<tr>
<td>Widgets</td>
<td>Includes the basic objects that you can insert on the design canvas.</td>
</tr>
<tr>
<td>OCX</td>
<td>Includes the OCX controls. The OCX controls are usually referred to as ActiveX controls.</td>
</tr>
<tr>
<td>SmartObjects</td>
<td>Includes the SmartObjects which allows you to create SmartObject instances.</td>
</tr>
</tbody>
</table>

**See also**

- Changing the Palette settings on page 660
- Types of objects on page 683
- Use Custom dialog on page 682
- Palette context menu on page 681

**Palette context menu**

Right-clicking on the Palette displays the Palette context menu. This menu has the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resize</td>
<td>Resizes the Palette pane. Select <strong>Resize</strong>, drag the Palette to resize it, and click to release.</td>
</tr>
<tr>
<td>Dock On</td>
<td>Determines the docking position of the Palette. You can dock the Palette to the right-side or left-side of the design canvas.</td>
</tr>
</tbody>
</table>
### Layout

Changes the layout of the controls in the Palette. The Layout options are:

- **Columns** - Displays the controls on the Palette in a column layout.
- **List** - Displays the controls on the Palette in a list layout.
- **Icons Only** - Displays the controls as icons on the Palette without the control text.
- **Details** - Displays the controls with a description on the Palette.

### Use Large Icons

Uses large icons for the controls on the Palette.

### Settings

Displays the Palette Settings dialog which allows you to set the Palette display settings.

### Use Custom

Displays the Use Custom dialog on page 682 which allows you to update the list of custom object files used by the GUI Designer.

---

**Note:** The Resize and Dock On options are available from the right-click context menu on the Palette title. The Layout, Use Large Icons, Settings, and Use Custom options are available from the right-click context menu on anywhere in the Palette.

---

### See also

- Changing the Palette settings on page 660
- Object Palette on page 680
- Use Custom dialog on page 682

### Use Custom dialog

The **Use Custom** dialog enables you to update the list of custom object files used by the GUI Designer.

The objects on the Palette refreshes when you add, modify, or remove a custom object file. The custom object files normally have a .cst extension in their filenames.

Access this dialog by clicking **Use Custom** from the right-click context menu on the Palette.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Custom Object Files</th>
<th>Displays the custom object files currently used by the GUI Designer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Displays the Add Custom Object File dialog which allows you to add a custom object file to the Custom Object Files list.</td>
</tr>
<tr>
<td>Modify</td>
<td>Displays the Modify Custom Object File dialog which allows you to modify the custom object file selected in the Custom Object Files list.</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes the selected custom object file from the Custom Object Files list.</td>
</tr>
<tr>
<td>Move Up</td>
<td>Moves the selected custom object file one position up in the Custom Object Files list.</td>
</tr>
<tr>
<td>Move Down</td>
<td>Moves the selected custom object file one position down in the Custom Object Files list.</td>
</tr>
</tbody>
</table>
### Types of objects

#### Basic static objects

The widgets section in the Palette contains the basic static ABL objects which you can add to the design canvas. Basic objects are usually referred to as user-interface components.

The Widgets section includes the following objects:

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Pointer" /></td>
<td>Enables you to select, rearrange, or resize objects on the design canvas.</td>
</tr>
</tbody>
</table>
| ![DB Fields](image) | Displays the Table Selector dialog on page 756 which allows you to insert database fields.  
**Note:** You are prompted to connect to a database, if the database connection is not configured with your current project. |
| ![Query](image) | Displays the Query Builder Dialog Box -- Table Mode on page 764 which allows you to insert a query.  
**Note:** You are prompted to connect to a database, if the database connection is not configured with your current project. |
| ![Browse](image) | Displays the Query Builder Dialog Box -- Table Mode on page 764. An object which displays the results of a database query. |
| ![Frame](image) | Inserts a frame. Use the frame to group other objects on the design canvas. |
| ![Rectangle](image) | Inserts a rectangle. Use rectangle to group other objects on the design canvas. |
| ![Image](image) | Inserts an image. |
| ![Radio Set](image) | Inserts a set of three radio buttons. You can change the number of buttons in the set after adding the object. |
| ![Toggle Box](image) | Inserts a check box. |
| ![Slider](image) | Inserts a slider control. You can select to adjust the widget vertically or horizontally. |
| ![Button](image) | Inserts a button. |
Selection List
Inserts a selection list. A Selection List presents a list scrollable set of items displayed as character strings.

Editor
Inserts a multi-line text editor.

Combo Box
Inserts a drop-down with a list of elements to display on the design canvas.

Fill In
Inserts a single-line text editor.

Text
Inserts a text field which holds static text.

For more information see, Types of objects on page 683

ActiveX objects
The OCX controls are usually referred to as ActiveX controls. These controls are self-contained units of functionality and content that are independent of the application in which they are used.

An ActiveX Control is defined in a special dynamic link library with an OCX extension or Object Component Extension. The ActiveX Controls include a wide variety of user interface components that are not available as native ABL widgets, such as spin box, enhanced combo box, and various dialog, meter, Internet-enabled, and picture controls.

The OCX section includes the following options:

| OCX | Displays the Choose Control dialog on page 755 which allows you to select an ActiveX control from the list. |
| CSSpin | Inserts an ActiveX spin control. Spin control allows you to set an Integer value. You can choose the upper or lower bounds of the range and the increment or decrement value. |
| CSCombo | Inserts an ActiveX combo box. |
| PSTimer | Inserts a timer that generates an event at some regular interval that you specify. |

For more information, see Types of objects on page 683

Note: Progress Developer Studio for OpenEdge on 64-bit Windows supports only the ActiveX control PSTimer. So Progress Software recommends that you continue to use the 32-bit version of Progress Developer Studio for OpenEdge if you extensively use other OCX controls such as CSCombo or CSList in your AppBuilder projects.

SmartObjects
The SmartObjects in the Palette allows you to create a SmartObject instance.
SmartObject instance is a copy of a SmartObject master running in memory. Many SmartObject instances can be created from a single SmartObject master file.

The SmartObjects section includes the following objects:
<table>
<thead>
<tr>
<th>SmartObject type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![SmartDataObject]</td>
<td>A SmartObject that holds a query and associated data update logic. It is supplied in two forms: a customizable template, and a pre-compiled dynamic object.</td>
</tr>
<tr>
<td>![DataView]</td>
<td>Inserts a DataView instance or a DataView master.</td>
</tr>
<tr>
<td>![Simple SmartObject]</td>
<td>A SmartObject that serves as a basis for new SmartObject types you might create.</td>
</tr>
<tr>
<td>![SmartFolder]</td>
<td>A SmartObject that holds folders to support a tab-folder application.</td>
</tr>
<tr>
<td>![SmartPanel]</td>
<td>A SmartObject that holds a related set of buttons, such as navigation buttons or update buttons.</td>
</tr>
<tr>
<td>![SmartDataBrowser]</td>
<td>A SmartObject that connects to a SmartDataObject and holds a browse object to view database records and to update them through the SmartDataObject.</td>
</tr>
<tr>
<td>![SmartDataViewer]</td>
<td>A SmartObject that holds a frame to preview and possibly update specific database fields. Database field values are obtained from a SmartDataObject or a SmartDataBrowser.</td>
</tr>
<tr>
<td>![SmartDataField]</td>
<td>A SmartObject that displays a single data field in a SmartDataViewer. There are two types of static SmartDataFields: SmartSelect and Time Field.</td>
</tr>
<tr>
<td>![SmartLOBField]</td>
<td>Inserts a SmartLOBField instance.</td>
</tr>
<tr>
<td>![SmartFilter]</td>
<td>A SmartObject that displays data fields in a frame, filtered by user-specified selection criteria.</td>
</tr>
<tr>
<td>![SmartToolBar]</td>
<td>A SmartObject that provides a toolbar and/or a menu interface to application commands.</td>
</tr>
<tr>
<td>![SmartContainer]</td>
<td>A SmartObject that contains other SmartObjects (including possibly other SmartContainers). There are four types of SmartContainer: SmartWindows, SmartDialogs, SmartFrames, and Simple SmartContainers.</td>
</tr>
<tr>
<td>![SmartBusinessObject]</td>
<td>A specialized container that can integrate up to 20 SmartDataObjects. It allows you to perform complex updates in a single server-side transaction.</td>
</tr>
<tr>
<td>![SmartProducer]</td>
<td>A message-interface object that creates and sends messages using the SonicMQ system.</td>
</tr>
<tr>
<td>![SmartConsumer]</td>
<td>A message-interface object that extracts incoming messages from the SonicMQ system and hands them off for processing.</td>
</tr>
<tr>
<td>![SmartRouter]</td>
<td>A utility object that routes incoming messages to the appropriate SmartB2BObject for processing.</td>
</tr>
<tr>
<td>![SmartB2BObject]</td>
<td>A specialized message-handling object. It creates or consumes formal, schema-based XML messages.</td>
</tr>
<tr>
<td>SmartObject type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>![SmartSender]</td>
<td>A messaging object that creates messages, possibly by transforming data in some way that you define.</td>
</tr>
<tr>
<td>![SmartReceiver]</td>
<td>A messaging object that consumes messages, possibly by transforming their content in some way that you define.</td>
</tr>
<tr>
<td>![SmartTreeview]</td>
<td>A SmartObject that provides a tree view control interface to display information hierarchically.</td>
</tr>
</tbody>
</table>

**Note:** You are prompted to connect to a database, if the database connection is not configured with your current project. When you click **OK**, the [Database Connections properties](#) page appears.

**See also**
- About SmartObjects on page 637
- Adding objects to the design window on page 648
- Types of objects on page 683
- Open SmartObject dialog on page 788

**Dynamic objects**

Progress Dynamics provides the following dynamic object types:

<table>
<thead>
<tr>
<th>Dynamic object type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dynamic SmartDataField objects</strong></td>
<td>This is a field object that contains data from a single database table, and displays the data for selection in a Dynamic SmartDataViewer.</td>
</tr>
<tr>
<td><strong>Dynamic Window Container objects</strong></td>
<td>This is a logical object that contains instances of predefined objects positioned on one or more pages with links for data communication. Use these objects to create your own window object types.</td>
</tr>
<tr>
<td><strong>Dynamic Field Container objects</strong></td>
<td>This is an object that contains field objects.</td>
</tr>
<tr>
<td><strong>Dynamic TreeView object</strong></td>
<td>This is a hybrid of a window container and field container object.</td>
</tr>
</tbody>
</table>

**Dynamic SmartDataField objects**

Progress Dynamics provides the following Dynamic SmartDataField objects:
Dynamic Combo SmartDataField
Displays a relatively small set of values in a drop-down list from which you can select a single value. Use this type of field object for 50 or fewer values.

Dynamic Lookup SmartDataField
Displays a large set of values in a separate browser from which you can filter a large number of values, select a single value, and return a value to the calling container. Use this type of field object for 50 or more values.

Use the **Dynamic SmartDataField Maintenance** dialog (OpenEdge > Build > Dynamic Development > SmartDataField > Maintenance option) to define dynamic SmartDataField objects.

**Dynamic Window Container objects**
Progress Dynamics provides the following dynamic window container objects:

<table>
<thead>
<tr>
<th>Dynamic Dependent Window (DynFold)</th>
<th>Contains one or more folder pages for managing data. Each folder page can contain one or more browsers, viewers, or toolbars, and can have one or more associated SDOs or SBOs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic Independent Window (DynObjc)</td>
<td>Contains standard menus, toolbars, and browsers used to view, filter, print, select, and export data. It typically contains a Dynamic SmartDataBrowser and a toolbar from which you can launch a Dynamic Dependent Window.</td>
</tr>
<tr>
<td>Dynamic Menu Controller (DynMenc)</td>
<td>Contains user-defined menus, toolbars, buttons, and status bar used to navigate an application and control application functions.</td>
</tr>
</tbody>
</table>

Use the **Container Builder** window (OpenEdge > Build > Container Builder) to build dynamic window container objects and maintain all of their constituent pages, object instances, links, menu structures, and properties. Use the **Toolbar and Menu Designer** window (OpenEdge > Build > Toolbar Menu Designer) to define your own toolbars and menu bars. Use the **Dynamic Launcher** dialog box to run a dynamic window container object (after you save it in the Repository).

**Dynamic Field Container objects**
Progress Dynamics provides the following dynamic field container objects:

<table>
<thead>
<tr>
<th>Dynamic SmartDataBrowser (DynBrow)</th>
<th>Provides visualization for DataFields from multiple rows of an SDO query.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic SmartDataObject (DynSDO)</td>
<td>Contains a query and associated data update logic. It can serve as a data source and a data target for other Dynamic SmartObjects such as Dynamic SmartDataBrowsers and Dynamic SmartDataViewers.</td>
</tr>
</tbody>
</table>

**Note:** A Dynamic SmartDataObject is a non-visual object

| Dynamic SmartDataViewer (DynView) | Provides visualization for DataFields from a single row of an SDO query. A viewer typically provides update access to data in a query. |
Dynamic TreeView object

A Dynamic TreeView (DynTree) object is a hybrid of a window container and field container object. It provides visualization for hierarchical data, from an SDO, which appears in a TreeView control.

A TreeView control displays a list of labeled nodes in a pane, each of which can contain a list of labeled child nodes that can be optionally displayed offset and under the parent node. Each node typically displays as an iconic image defined to indicate the type of data identified by the node. A user typically navigates the TreeView using a pointer device to expand and collapse the child nodes. The result of navigating a TreeView is typically displayed in a separate pane, usually to the right of the TreeView pane.

Use the Tree Node Control window (OpenEdge > Build > Tree > Node Control) and the TreeView Wizard Control window (OpenEdge > Tools > Dynamics Development > TreeView Wizard) to define and assemble a hierarchy of nodes for a Dynamic TreeView object, and then generate the object.

**Note:** To modify a Dynamic TreeView object, you must delete the existing definition of the object, make your modifications, and generate a new one.

---

**See also**

About Progress Dynamics on page 640
Working with Progress Dynamics on page 662

---

### Toolbar and menu options

#### GUI Designer toolbar

GUI Designer toolbar contain actions that apply only to the ABL procedure (.w) files or GUI Designer perspective. The toolbar are available and enabled only when the AppBuilder perspective or an ABL GUI procedure (.w) file is opened.

**Note:**

You can use the Customize Perspective dialog to show or hide the toolbar buttons which are visible in your perspective. Access this dialog by selecting Windows menu > Customize Perspective.

The toolbar contains the following buttons:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Procedure Settings" /></td>
<td>Opens the Procedure Settings dialog box on page 700 for the current procedure file. This dialog allows you to modify characteristics for the current procedure.</td>
</tr>
<tr>
<td><img src="image" alt="View Source (F9)" /></td>
<td>Opens the source editor for the current design window.</td>
</tr>
<tr>
<td><img src="image" alt="Property Sheet" /></td>
<td>Opens the Property Sheet dialog for a selected object. This dialog allows you to define and modify characteristics for the selected object.</td>
</tr>
</tbody>
</table>
Option | Description
--- | ---
[ ] Dynamic Properties | Opens the Dynamic Properties window which allows you to edit repository-based attributes and events for one or more master objects or object instances registered in the Repository.

**Note:** The Dynamic Properties window opens and runs in a window outside of the Progress Developer Studio Eclipse framework. Online help is available from the utility's Help button or F1 key.

[ ] Choose Color | Opens the Choose Color dialog on page 753 which allows you to set foreground and background colors for the selected objects in the design window.

[ ] Go To Page | Allows you to change the current design page. You can go to the selected design page from the current design page.

This option is enabled when the current design window is a SmartContainer. See Paging in SmartObjects on page 638.

**See also**

- AppBuilder perspective components on page 676
- Showing or hiding toolbar buttons and menu items on page 669

**GUI Designer Dynamics toolbar**

GUI Designer Dynamics toolbar contain actions that apply only to the ABL procedure (.w) files or GUI Designer perspective. The toolbar will be available and enabled only when the AppBuilder perspective or an ABL procedure file is opened.

The toolbar contains the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ ] OpenObject</td>
<td>Opens an existing object from the Open Object dialog. You can also view and object's properties or remove an object from the Repository.</td>
</tr>
<tr>
<td>[ ] DynamicsProperties</td>
<td>Displays the Dynamics Properties dialog to edit the properties of a selected object.</td>
</tr>
</tbody>
</table>

**Layout menu options**

Use the Layout menu to change the layout, order, and alignment of basic objects and SmartObject instances in a frame. The Layout menu is available only when the AppBuilder perspective or an ABL procedure (.w) file is opened.

The Layout menu includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alternate Layout</td>
<td>Allows you to change to an alternate layout.</td>
</tr>
<tr>
<td>Center Left-to-Right in Frame</td>
<td>Aligns the selected object horizontally in a frame.</td>
</tr>
</tbody>
</table>
Aligns the selected object horizontally in a frame.

Adjusts the horizontal spacing of the selected objects.

Adjusts the vertical spacing of the selected objects.

Makes the selected basic object or SmartObject instance overlay other basic objects or SmartObject instances of its type.

Makes the selected basic object or SmartObject instance underlay other basic objects or SmartObject instances of its type.

Aligns the selected basic object or SmartObject instances within the Design window.

Allows you to specify the tab order of objects in a frame.

Opens the Customization Priority Editor. The Customization priority menu option is available with only Dynamics.

Allows you to change the current design page. You can go to a selected design page from the current design page. This option is enabled when the current design window is a SmartContainer object. For more information, see Goto Page dialog on page 780.

See also

Selecting standard or alternate layout on page 652
Tab Editor dialog on page 699

Progress Dynamics menu options

The Progress Dynamics actions are now available from within the Progress Developer Studio for OpenEdge. These dynamics actions enable you to develop dynamics applications in an OpenEdge project.

The Progress Dynamics actions are available only with the Dynamics projects. See Setting Progress Dynamics projects.

Note:

You can use the Customize Perspective dialog to show or hide the Progress Dynamics menu actions which are visible in your perspective. Access this dialog by selecting Windows > Customize Perspective.

The Dynamics actions available from the File menu are:

<table>
<thead>
<tr>
<th>Dynamics action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️ Open Object</td>
<td>Opens the Open Object dialog that enables you to browse for objects stored in the Repository and open them for editing in the GUI Designer. See Open Object dialog on page 808.</td>
</tr>
</tbody>
</table>
### Dynamics action

<table>
<thead>
<tr>
<th>Dynamics action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Associated Procedure</td>
<td>Opens an associated procedure if the currently selected object has a custom procedure or a data logic procedure associated with it.</td>
</tr>
</tbody>
</table>
| Save Dynamic Object As Static    | Opens the **Save As Static** dialog that enables you to save the following dynamic objects as static objects:  
  - Dynamic SmartDataViewers (DynView)  
  - Dynamic SmartDataObjects (DynSDO)  
  See Saving dynamic objects as static on page 665.  
  **Note:** This menu action is available when the type of object is a dynamic object. |
| Save Static Object As Dynamic    | Opens the **Save As Dynamic Object** dialog that enables you to save the following static objects as dynamic objects:  
  - Static SmartDataViewers (StaticSDV)  
  - Static SmartDataBrowsers (StaticSDB)  
  - Static SmartDataObjects (SDO)  
  See Saving static objects as dynamic on page 667.  
  **Note:** This menu action is available when the type of object is a static object. |
| Register in Repository           | Opens the **Register in Repository** dialog that enables you to register an existing static object in the repository.  
  See Registering objects in Repository on page 668.  
  **Note:** This menu action is enabled when the type of object is a static (.w) object and is not already registered in the repository. |

The Dynamics actions available from the **OpenEdge > Tools** menu are:

<table>
<thead>
<tr>
<th>Dynamics action</th>
<th>Description</th>
</tr>
</thead>
</table>
| Dynamics Administration          | Opens the **Progress Dynamics Administration** dialog that provides access to site management and application development tools.  
  This dialog includes the following elements:  
  - **Menu** - provides access to menus for administration commands.  
  - **Toolbar** - provides access to commonly used administration commands and tools.  
  - **Status Area** - displays session status information such as the user currently logged in, company, and current date. |
## Dynamics action

<table>
<thead>
<tr>
<th>Dynamics action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamics Development</td>
<td>Opens the Progress Dynamics Development dialog that enables you to create and edit Repository object definitions and perform related development tasks.</td>
</tr>
<tr>
<td></td>
<td>This dialog includes the following elements:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Menu</strong> - provides access to menus for development commands.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Toolbar</strong> - provides access to commonly used development commands and tools.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Status Area</strong> - displays session status information such as user currently logged in, company, and current date.</td>
</tr>
<tr>
<td>Dynamic Launcher</td>
<td>Opens the Dynamic Launcher dialog on page 812 that lets you run a dynamic object (after you save it in the Object Repository).</td>
</tr>
<tr>
<td>Clear Repository Cache</td>
<td>Clears the client-side cache to run the latest version of an object.</td>
</tr>
</tbody>
</table>

The Dynamics actions available from the OpenEdge > Build menu are:

<table>
<thead>
<tr>
<th>Dynamics action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Generator</td>
<td>Opens the Object Generator dialog that enables you to generate various types of objects for one or more tables in a database. You can generate objects by selecting the following: dynamic SDOs, datalogic procedures, dynamic browsers, dynamic viewers, and data fields. You can also set some default characteristics for the objects.</td>
</tr>
<tr>
<td>Container Builder</td>
<td>Opens the Container Builder dialog that enables you to build dynamic container objects and maintain all of their pages, object instances, links, menu structures, and properties.</td>
</tr>
<tr>
<td>Toolbar and Menu Designer</td>
<td>Opens the Toolbar and Menu Designer dialog that enables you to build custom menus and toolbars for your application.</td>
</tr>
<tr>
<td>Class Maintenance</td>
<td>Opens the Class Maintenance dialog that enables you to maintain object classes, their relationship to other object classes, and their related data, including attributes, UI events, and supported links.</td>
</tr>
</tbody>
</table>

**Note:** Appropriate use of this tool requires an advanced knowledge of classes and their maintenance and how changes affect the way objects function.
<table>
<thead>
<tr>
<th>Dynamics action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository Maintenance</td>
<td>Opens the <strong>Repository Maintenance</strong> dialog that enables you to maintain dynamic objects and their relationship to other objects directly in the Repository using a TreeView interface.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Appropriate use of this dialog requires an advanced knowledge of the Repository, its representation of dynamic objects, and changes to the data affect the way objects function.</td>
</tr>
<tr>
<td>SmartDataField Maintenance</td>
<td>Opens the <strong>SmartDataField Maintenance</strong> dialog that enables you to create and edit Dynamic Lookup and Dynamic Combo objects. You specify property values for Dynamic Lookup instances in the Dynamic Lookup tab, and for Dynamic Combo instances in the Dynamic Combo tab.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> After you specify property values for the SmartDataField object, save the object to register its properties in the Repository.</td>
</tr>
<tr>
<td>Tree Node Control</td>
<td>Opens the <strong>Tree Node Control</strong> dialog that enables you to create and edit a node in the hierarchy of nodes for a Dynamic TreeView object.</td>
</tr>
<tr>
<td>Dynamic TreeView Builder</td>
<td>Opens the <strong>Dynamic TreeView Builder</strong> dialog that enables you to create and manage repository-object data for Dynamic TreeView containers. The TreeView Builder maintains the information required to build the objects in a Dynamic TreeView container.</td>
</tr>
<tr>
<td>Set Site Number</td>
<td>Opens the Set Site Number dialog that enables you to set a unique site identification number for a registered Progress Dynamics site in the Progress Dynamics Repository Database (icfdb.db).</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> A site identification number uniquely identifies a Progress Dynamics Repository. More importantly, it ensures the integrity of Repository data among Progress Dynamics sites and applications, worldwide, as well as between versions of Progress Dynamics.</td>
</tr>
</tbody>
</table>

For more information on Progress Dynamics, see Progress Dynamics manuals in the **OpenEdge Product Documentation** section of the Progress Communities web site.

**See also**
- *Showing or hiding toolbar buttons and menu items* on page 669
- *Progress Dynamics dialogs* on page 808

**GUI Designer context menu**

Right-clicking on the design window displays the GUI Designer context menu. This menu has the following options:
<table>
<thead>
<tr>
<th>Menu option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undo</td>
<td>Undo the last action performed on the design window.</td>
</tr>
<tr>
<td>Cut</td>
<td>Cut the selected widget.</td>
</tr>
<tr>
<td>Copy</td>
<td>Copy the selected widget.</td>
</tr>
<tr>
<td>Paste</td>
<td>Paste the selected widget.</td>
</tr>
<tr>
<td>Duplicate</td>
<td>Create a copy of the selected widget.</td>
</tr>
<tr>
<td>Delete</td>
<td>Delete a selected widget.</td>
</tr>
<tr>
<td>View Source</td>
<td>Open the source code editor.</td>
</tr>
<tr>
<td>Run</td>
<td>Run the current AppBuilder procedure (.w) file.</td>
</tr>
<tr>
<td>Compile</td>
<td>Compile the current AppBuilder procedure (.w) file.</td>
</tr>
<tr>
<td>Add Function</td>
<td>Open the <strong>Add Function</strong> wizard to add a function to the current procedure.</td>
</tr>
<tr>
<td>Add Procedure</td>
<td>Open the <strong>Add Procedure</strong> wizard to add an internal procedure to the current procedure.</td>
</tr>
<tr>
<td>Add Trigger</td>
<td>Open the <strong>Add Trigger</strong> wizard to add a trigger for a selected object.</td>
</tr>
<tr>
<td>Copy to File</td>
<td>Open the <strong>Copy To File</strong> dialog which allows you to copy basic objects and SmartObject instances and their associated internal procedures, functions, and triggers to an export (.wx) file.</td>
</tr>
<tr>
<td>Insert from File</td>
<td>Opens the <strong>Choose From File</strong> dialog which allows you to locate and insert a particular filename or image.</td>
</tr>
<tr>
<td>Tab Order</td>
<td>Open the <strong>Tab Editor</strong> dialog which allows you to change the tab order of the widgets or objects present on the current design window.</td>
</tr>
<tr>
<td>Goto Page</td>
<td>Open the <strong>Goto Page</strong> dialog which provides the ability to organize sets of SmartObjects into logical pages.</td>
</tr>
</tbody>
</table>

**See also**

- Creating ABL GUI procedure files on page 644
- Opening ABL GUI procedure files on page 645

**SmartObjects context menu**

Right-clicking a SmartObject instance on the design window displays SmartObject context menu. This menu has the following options:
<table>
<thead>
<tr>
<th>Menu option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties</td>
<td>Open the <strong>Property Sheet</strong> dialog to edit the attributes of the selected SmartObject instance.</td>
</tr>
<tr>
<td>Instance Properties</td>
<td>Open the <strong>SmartObject instance properties</strong> dialog to view the available properties and their corresponding values.</td>
</tr>
<tr>
<td>SmartInfo</td>
<td>Open the <strong>SmartInfo</strong> dialog to view SmartLink information supported by the selected SmartObject instance.</td>
</tr>
<tr>
<td>SmartLinks</td>
<td>Open the <strong>SmartLinks</strong> dialog to view, add, modify, and remove SmartLinks between SmartObject instances in the current design window.</td>
</tr>
<tr>
<td>Edit Master</td>
<td>Open the SmartObject master in the design window to edit the SmartObject master.</td>
</tr>
<tr>
<td>Delete Instance</td>
<td>Delete a selected SmartObject instance.</td>
</tr>
</tbody>
</table>

**See also**
- Inserting SmartObjects on page 669
- Choose SmartObject dialog on page 788

**PRO*Tools utilities**

PRO*Tools is now available from within the Progress Developer Studio for OpenEdge. PRO*Tools is a set of utility programs which help you develop and run OpenEdge applications. These utilities are available on the toolbar and from the menu (**OpenEdge > General ProTools**) in your perspective.

You can use the **Customize Perspective** dialog to show or hide the PRO*Tools menu items and toolbar buttons which are visible in your perspective. Access this dialog by selecting **Windows > Customize Perspective**.

The PRO*Tools utilities are grouped into General, AppBuilder, and ADM2 categories.

**General PRO*Tools utilities**

The General PRO*Tools utilities are available with all the OpenEdge perspectives. These utilities are available on the toolbar and from the menu (**OpenEdge > General PRO*Tools**) in your perspective.

**Note:** Each of these PRO*Tools utilities opens and runs in a window outside of the Progress Developer Studio Eclipse framework. Online help is available from the utility's menu bar or Help button.

The General PRO*Tools include the following utilities:

<table>
<thead>
<tr>
<th>General PRO*Tools utility</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config Info</td>
<td>Opens the <strong>Configuration File</strong> dialog that lists all the Progress product installation information and their corresponding serial numbers.</td>
</tr>
<tr>
<td>Color Changer</td>
<td>Opens the <strong>Color Changer</strong> dialog that enables you to change the Progress system colors 0 through 15.</td>
</tr>
<tr>
<td>General PRO*Tools utility</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>OS Prompt</td>
<td>Opens a new MS-Windows DOS console.</td>
</tr>
<tr>
<td>Procedure Viewer</td>
<td>Opens the <strong>Procedure Object Viewer</strong> that displays a list of running persistent procedures in the current session. This viewer also displays a list of any super procedures for the selected persistent procedure. Use the <strong>Procedure Object Viewer</strong> when you debug an application to verify the existence of and to edit persistent procedures and super procedures and test their internal methods.</td>
</tr>
<tr>
<td>Propath Search</td>
<td>Opens the <strong>Propath File Search</strong> dialog that allows you to specify a file you want to locate in Propath. Progress displays the full path of each occurrence in Propath order. Click <strong>Open File</strong> to open any source file from the search result. (r-code file cannot be opened.)</td>
</tr>
<tr>
<td>Session Settings</td>
<td>Opens the <strong>Session Attributes</strong> dialog that lists all the attributes and their corresponding values for their Progress SESSION handle.</td>
</tr>
<tr>
<td>Screen Capture</td>
<td>Opens the <strong>Screen Capture</strong> dialog that lets you capture screen images of the Progress windows or the complete screen. You can then edit, save it in a file, or print the image.</td>
</tr>
<tr>
<td>Protogen</td>
<td>Opens the <strong>Prototype Generator</strong> dialog that generates an include file for the specified super procedure containing the prototypes of all internal procedures and user-defined functions.</td>
</tr>
<tr>
<td>PRO*Spy</td>
<td>Opens the <strong>PRO*Spy Plus</strong> dialog. PRO*Spy Plus is a debugging and diagnostic tool used to track call stack information from within an ABL application.</td>
</tr>
<tr>
<td>WebClient Application Assembler</td>
<td>Opens the <strong>WebClient Application Assembler</strong> dialog that enables you to define and assemble the client of an ABL application for deployment as a version controlled WebClient application.</td>
</tr>
<tr>
<td>WebClient Deployment Packager</td>
<td>Opens the <strong>WebClient Deployment Packager</strong> dialog that enables you easily modify the deployment configuration of an application that you have previously generated with the WebClient Application Assembler. It extracts information from an existing .prowcapc file and makes it editable.</td>
</tr>
</tbody>
</table>

**See also**
- Adding user-defined utilities on page 669
- Showing or hiding toolbar buttons and menu items on page 669
- PRO*Tools utilities on page 695
AppBuilder PRO*Tools utilities

The AppBuilder PRO*Tools utilities are available only with the AppBuilder perspective. These utilities are available on the toolbar and from the menu (OpenEdge > AppBuilder PRO*Tools) in your AppBuilder Perspective.

Note: Each of these PRO*Tools utilities opens and runs in a window outside of the Progress Developer Studio Eclipse framework. Online help is available from the utility’s menu bar or Help button.

The AppBuilder PRO*Tools include the following utilities:

<table>
<thead>
<tr>
<th>AppBuilder PRO*Tools utility</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>📑 Screen Scaling Utility</td>
<td>Opens the <strong>Screen Scaling</strong> window that assists in scaling the size and</td>
</tr>
<tr>
<td></td>
<td>changing the position of objects in AppBuilder-generated procedures.</td>
</tr>
<tr>
<td>✗ COM Object Viewer</td>
<td>Opens the <strong>COM Object Viewer</strong> window. The COM Object Viewer displays</td>
</tr>
<tr>
<td></td>
<td>the contents of Type Libraries and the Progress syntax you would use to</td>
</tr>
<tr>
<td></td>
<td>programmatically create ActiveX automation objects and controls and</td>
</tr>
<tr>
<td></td>
<td>manipulate their methods, properties, and events.</td>
</tr>
<tr>
<td>🏧 Static Widget ID Assignment</td>
<td>Opens the <strong>Widget ID Assignment Utility</strong> window that allows you to assign</td>
</tr>
<tr>
<td></td>
<td>widget IDs to multiple containers without opening them individually. Widget</td>
</tr>
<tr>
<td></td>
<td>IDs are application-defined IDs for widgets or objects that aid testing with</td>
</tr>
<tr>
<td></td>
<td>third-party tools. With the WIDGET-ID attribute, you can specify a unique ID</td>
</tr>
<tr>
<td></td>
<td>for a widget. The unique ID provides a means for third-party testing tools</td>
</tr>
<tr>
<td></td>
<td>to consistently identify the widget across sessions. You can use the AppBuilder</td>
</tr>
<tr>
<td></td>
<td>Preferences dialog to enable the AppBuilder to automatically add widget IDs</td>
</tr>
<tr>
<td></td>
<td>to widgets as they are created. See AppBuilder preferences on page 677.</td>
</tr>
<tr>
<td>✸ Runtime Widget ID Assignment</td>
<td>Opens the <strong>Runtime WIDGET-ID Assignment Tool</strong> window that allows you to</td>
</tr>
<tr>
<td></td>
<td>assign widget IDs in ADM2 and Progress Dynamics applications that cannot</td>
</tr>
<tr>
<td></td>
<td>store widget IDs in their code as static widgets.</td>
</tr>
<tr>
<td></td>
<td>Use the Runtime Widget ID Assignment utility to assign widget IDs to the</td>
</tr>
<tr>
<td></td>
<td>following:</td>
</tr>
<tr>
<td></td>
<td>• Widgets in dynamic objects, for example, a dynamic SmartDataViewer.</td>
</tr>
<tr>
<td></td>
<td>• Dynamically-rendered widgets in SmartObjects, for example, a SmartSelect.</td>
</tr>
<tr>
<td></td>
<td>• Multiple instances of a SmartObject in the same run-time container.</td>
</tr>
</tbody>
</table>

See also

- Adding new PRO*Tools utilities on page 669
- Showing or hiding toolbar buttons and menu items on page 669
- PRO*Tools utilities on page 695
ADM2 PRO*Tools utilities

The ADM2 PRO*Tools utilities are available only with the AppBuilder Perspective. These utilities are available on the toolbar and from the menu (OpenEdge > ADM2 PRO*Tools) in your AppBuilder Perspective.

**Note:** Each of these PRO*Tools utilities opens and runs in a window outside of the Progress Developer Studio Eclipse framework. Online help is available from the utility's menu bar or Help button.

The ADM2 PRO*Tools include the following utilities:

<table>
<thead>
<tr>
<th>ADM2 PRO*Tools utility</th>
<th>Description</th>
</tr>
</thead>
</table>
| ![SmartObject Upgrade Utility](image) | Opens the **ADM1 to ADM2 SmartObject Conversion Utility** window that facilitates the process of upgrading (loading and saving) SmartObjects from version 8.1 to 9.0 and later.  
**Note:** The application database must be connected for you to use this utility. |
| ![Service Parameter Maintenance](image) | Opens the **Service Parameter Maintenance** window that enables you to define instances of application partitions that you can implement in AppBuilder procedures. The tool supports two types of application partitions:
- AppServer partitions
- Java Message Service (JMS) Server partitions |
| ![ADM to Dynamics](image) | Opens the **Static SmartObject to Dynamic Object Migration Utility** window. This utility facilitates the process of converting static SmartObjects to dynamic objects as follows:
- Progress Version 8, Progress Version 9, and Progress Dynamics Version 1.1A static viewers to dynamic viewers.
- Progress Version 8 and Version 9 browsers to dynamic browsers.
- Progress Version 9 SmartDataObjects to dynamic SmartDataObjects.
- Progress Version 9 SmartBusinessObjects to dynamic SmartBusinessObjects.
You can use this utility to migrate entire batches of objects from their ABL form to their dynamic form.  
**Note:** You must make connections to the icfdb repository database and your application databases before you can use this utility. |
| ![XML Schema Mapping](image) | Opens the **XML Schema Mapping** window that enables you to create and update schema-based XML mapping files for the use of the SmartB2BObject. |

**See also**

Adding user-defined utilities on page 669
GUI Designer dialogs

AppBuilder dialogs

Tab Editor dialog

The Tab Editor dialog includes the following options:

<table>
<thead>
<tr>
<th>Tabbing Options</th>
<th>Default is the default tabbing option for all frames.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Note:</strong> The Move First, Move Up, Move Down, and Move Last options are not available when you select this option.</td>
</tr>
<tr>
<td></td>
<td>Custom allows tabbing using the Tab Editor Move buttons.</td>
</tr>
<tr>
<td>Left-To-Right By Column</td>
<td>allows tabbing to move top to bottom in each column from left to right. Columns are identified by the left-most coordinates of each tabbable object.</td>
</tr>
<tr>
<td>Left-To-Right By Row</td>
<td>allows tabbing to move left to right in each row from top to bottom.</td>
</tr>
<tr>
<td>Right-To-Left By Column</td>
<td>allows tabbing to move top to bottom in each column from right to left. Columns are identified by the right-most coordinates of each tabbable object.</td>
</tr>
<tr>
<td>Right-To-Left By Row</td>
<td>allows tabbing to move right to left in each row from top to bottom.</td>
</tr>
<tr>
<td>Tabbable Objects</td>
<td>Lists the objects in the currently selected frame that you can access with the TAB key.</td>
</tr>
<tr>
<td>Move First</td>
<td>Moves the selected object to the top of the tabbable objects list and makes it the first object in the tab order for the frame.</td>
</tr>
<tr>
<td>Move Up</td>
<td>Moves the selected object up one place in the tabbable objects list.</td>
</tr>
<tr>
<td>Move Down</td>
<td>Moves the selected object down one place in the tabbable objects list.</td>
</tr>
<tr>
<td>Move Last</td>
<td>Moves the selected object to the bottom of the tabbable objects list and makes it the last object in the tab order for the frame.</td>
</tr>
</tbody>
</table>

See also

- Setting tab order on page 653
- Working with objects on page 647
**Procedure Settings dialog box**

Use the **Procedure Settings** dialog to modify certain characteristics of a GUI Designer procedure, which is a special class of an OpenEdge procedure generated by the AppBuilder. AppBuilder procedures include window procedures, SmartObject masters, method libraries, structured includes, and structured procedures.

Access the dialog by clicking the **Procedure Settings** icon on the GUI Designer toolbar.

The **Procedure Settings** dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Displays the type of the selected procedure.</td>
</tr>
<tr>
<td>File Name</td>
<td>Displays the filename of the selected procedure.</td>
</tr>
<tr>
<td>Description</td>
<td>Displays a description of the selected procedure.</td>
</tr>
<tr>
<td>OCX Binary</td>
<td>Select a file to use as the OCX binary file.</td>
</tr>
<tr>
<td>Compile in</td>
<td>Select the directory where the GUI Designer saves the compiled (.rcode) version of the current procedure file. By default, the compiled file is saved in the same directory as the procedure file.</td>
</tr>
<tr>
<td>External Tables</td>
<td>Displays the external tables associated with the procedure. You can modify this declaration.</td>
</tr>
<tr>
<td>Add/Modify</td>
<td>Displays the Table Selector dialog on page 756 that allows you to add an external table to the current procedure.</td>
</tr>
<tr>
<td>Run Persistent from AppBuilder</td>
<td>Select this option to run the procedure persistently.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> You must deactivate this toggle for a SmartDialog and the main window of an application.</td>
</tr>
<tr>
<td>AppServer Aware</td>
<td>Select this option for GUI procedures that need to run procedures on the AppServer. This generates at the top of the .w file an include file and declarations of AppServer handles that facilitate the execution of partitions defined in the Service Parameter Maintenance PRO*Tool. The include file registers adecomm/as-utils .w as a super procedure of the .w file. This makes the utilities appServerConnect and appServerDisconnect available to the procedure file.</td>
</tr>
<tr>
<td>Compile on Save</td>
<td>Select this option for the GUI Designer to automatically generate the .rcode file when compiling the current procedure. By default, this option is selected for all new master files based on the basic templates for windows, dialog boxes, and structured procedures.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Activating this toggle improves AppBuilder performance when running SmartObjects, because the AppBuilder does not have to compile the master file to create an instance.</td>
</tr>
<tr>
<td>Method Libraries</td>
<td>Opens the Method Libraries dialog on page 780 that allows you to add, remove, or modify the procedure's method libraries.</td>
</tr>
</tbody>
</table>
### Custom Lists
Opens the [Custom Lists dialog box](#) on page 786 that allows you to change the names of the preprocessor lists.

### Temp-Table Maintenance
Opens the [Temp-Table Maintenance Dialog Box](#) on page 783.

### SmartLinks
Opens the [SmartLinks dialog](#) on page 805 that allows you to modify the SmartLinks associated with the selected procedure.

**Note:** This option is only available when working with a SmartObject procedure.

### Pages
Opens the [Pages dialog](#) that allows you to control on which page in the procedure is displayed.

### Declarative Statements
Opens the [Edit Declarative Statements dialog](#). Use this dialog to add statements (such as, USING and ROUTINE LEVEL ON ERROR UNDO, and THROW) that must come before any definition and executable statement for proper compilation. These statements are generated into a new section in the source code. This section is written before both the optional temp-table definitions and the optional AppServer include.

### Advanced
Opens the [Advanced Procedure Settings dialog](#) that allows you to edit additional settings for the procedure.

For more information, see [GUI Designer toolbar](#) on page 688

### Advanced Procedure Settings Dialog Box
The [Advanced Procedure Settings dialog](#) is used to set advanced procedure settings for a GUI Designer procedure file.

You can access this dialog by clicking Advanced in the Procedure Settings dialog.

The Advanced Procedure Settings dialog includes the following options:

<table>
<thead>
<tr>
<th>Procedure Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Procedure Type</td>
<td>Select a procedure type for the current procedure. Choose a procedure type from the list.</td>
</tr>
<tr>
<td>New</td>
<td>Select this option to add a new procedure type.</td>
</tr>
<tr>
<td>Window</td>
<td>Select this option to allow the addition of a window to the procedure.</td>
</tr>
<tr>
<td>Frames</td>
<td>Select this option to allow the addition of frames to the procedure.</td>
</tr>
<tr>
<td>One Only</td>
<td>Select this option to allow the addition of only one frame to the procedure. This option is enabled only when Frames is selected.</td>
</tr>
<tr>
<td>Multiple</td>
<td>Select this option to allow the addition of multiple frames to the procedure. This option is enabled only when Frames is selected.</td>
</tr>
<tr>
<td>Smart Objects</td>
<td>Select this option to allow the addition of SmartObjects to the procedure.</td>
</tr>
<tr>
<td>Basic Objects</td>
<td>Select this option to allow the addition of basic objects to the procedure.</td>
</tr>
</tbody>
</table>
Database Fields | Select this option to allow the addition of database fields to the procedure.
---|---
Browses | Select this option to allow the addition of browses to the procedure.
Queries | Select this option to allow the addition of queries to the procedure.
Supported SmartLinks | Displays a list of the SmartLink types which the procedure currently supports.
Add | Displays the Add Supported Link dialog that allows you to add a SmartLink type to the Supported SmartLinks list.
Remove | Allows you to remove the SmartLink type currently selected in the Supported SmartLinks list.
Add fields to | This section has the following options:
  • Frame Query - Select this option to add new fields to the frame query.
  • External Tables - Select this option to add new fields to external tables.
  • Neither - Select this option if you do not want to enable addition of new fields to the frame query or external tables.
File Type | Select the file type for the procedure. Valid file types are .w, .i, and .p.
Template | Select this option to change a Master file to a template; clear this option to change a template to a Master file.
Run option | Select the Persistent Only option to specify that the procedure can only be run persistently.

For more information, see Procedure Settings dialog box on page 700

**Property Sheet dialogs**

The Property Sheet dialog allows you to modify the properties of a selected object. Access this dialog by selecting the object on the design canvas, and clicking Property Sheet icon on the GUI Designer toolbar.

**Property Sheet for Window**

The Property Sheet - Window dialog allows you to modify the window properties.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Object</th>
<th>Enter a new name for the window.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Enter a new title for the window.</td>
</tr>
<tr>
<td><strong>Icon Image</strong></td>
<td>Displays the Choose Image dialog which allows you to select an image .ico file. Enter the filename of the 32x32 icon for the window when minimized. For MS-Windows this can only be a .ico file.</td>
</tr>
<tr>
<td><strong>Small Icon Image</strong></td>
<td>Displays the Choose Image dialog which allows you to select an image. Enter the filename of the 16x16 icon for the window when minimized.</td>
</tr>
</tbody>
</table>
| **Column** | Displays the column value (in character units) where the window is placed from upper left corner on the design canvas.  
**Note:** This option is enabled when the Explicit Position check box is selected in the Other Settings section. |
| **Row** | Displays the row value (in character units) where the window is placed from upper left corner on the design canvas.  
**Note:** This option is enabled when the Explicit Position check box is selected in the Other Settings section. |
| **Width** | Enter a value (in character units) to specify the width of the window. |
| **Height** | Enter a value (in character units) to specify the height of the window. |
| **Virtual Width** | Enter the maximum and virtual width (in character units) of the window. |
| **Virtual Height** | Enter the maximum and virtual height (in character units) of the window. |
| **Other Settings** | This section allows you to select run-time attributes and other ABL related options. |
| **Color** | Displays the Choose Color dialog which allows you to change window background and foreground colors. |
| **Font** | Displays the Choose Font dialog which allows you to change the font displayed in the window status bar. |
| **Property Sheet - Popup Menu** dialog which allows you to create a pop-up menu for the window. |
| **Property Sheet - String Attributes** dialog which allows you to enter translation attributes for the window title. |
| **Property Sheet - Menu Bar** dialog which allows you to create a menu bar for the window. |
| **Property Sheet - Advanced Properties** dialog which allows you to edit advanced properties for the selected window. |

For more information, see Property Sheet dialogs on page 702

**Property Sheet for Dialog Box**

The **Property Sheet - Dialog Box** dialog allows you to modify the properties of dialog box.

This dialog includes the following options:

<p>| <strong>Object</strong> | Enter a new name for the dialog box. |
| <strong>Title</strong> | Enter a new title for the dialog box. |
| <strong>Query</strong> | Displays the database query for the dialog box. You can edit the query by clicking Query, which opens the Query Builder dialog. |
| <strong>Column</strong> | Displays the column value (in character units) where the dialog box is placed from upper left corner on the design canvas. |
| <strong>Note:</strong> This option is enabled when the Explicit Position check box is selected in the Other Settings section. This attribute only takes effect at run time. |
| <strong>Row</strong> | Displays the row value (in character units) where the window is placed from upper left corner on the design canvas. |
| <strong>Note:</strong> This option is enabled when the Explicit Position check box is selected in the Other Settings section. This attribute only takes effect at run time. |
| <strong>Width</strong> | Enter a value (in character units) to specify the width of the dialog box. |
| <strong>Height</strong> | Enter a value (in character units) to specify the height of the dialog box. |
| <strong>Other Settings</strong> | This section allows you to select run-time attributes and other ABL related options or accept the defaults. |</p>
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td>Displays the Choose Color dialog which allows you to change dialog box background color and foreground color for field-level objects in the dialog box.</td>
</tr>
<tr>
<td>Font</td>
<td>Displays the Choose Font dialog which allows you to change the font for field-level objects in the dialog box.</td>
</tr>
<tr>
<td>Popup Menu</td>
<td>Displays the Property Sheet - Popup Menu dialog which allows you to create a pop-up menu for the dialog box.</td>
</tr>
<tr>
<td>Translation Attributes</td>
<td>Displays the String Attributes dialog which allows you to enter translation attributes for the dialog box title.</td>
</tr>
<tr>
<td>Tab Order</td>
<td>Displays the Tab Editor dialog which allows you to change the tab order of objects in the dialog.</td>
</tr>
<tr>
<td>Advanced</td>
<td>Displays the Advanced Properties dialog which allows you to edit advanced properties for the selected dialog-box.</td>
</tr>
</tbody>
</table>

For more information, see Property Sheet dialogs on page 702

**Property Sheet for Menu Bar or Popup Menu**

The Property Sheet - Menu Bar or Popup Menu dialog allows you to specify the attributes while creating a menu bar and popup menu.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Menu</td>
<td>Enter a name for the menu.</td>
</tr>
<tr>
<td>Delete Menu</td>
<td>Allows you to delete the menu from the design window.</td>
</tr>
<tr>
<td>Title</td>
<td>Enter a title for the menu.</td>
</tr>
<tr>
<td>Label</td>
<td>Enter a label for the menu element.</td>
</tr>
<tr>
<td>: &lt;fill-in&gt;</td>
<td>Enter valid character-string attributes for the menu element label.</td>
</tr>
<tr>
<td>Object</td>
<td>Enter a name for the menu element.</td>
</tr>
<tr>
<td>Accelerator</td>
<td>Displays the accelerator key bound to the menu.</td>
</tr>
<tr>
<td>Key</td>
<td>Opens the Record Menu Accelerators dialog on page 781 which allows you to define an accelerator key to bind to the menu element.</td>
</tr>
<tr>
<td>Clear</td>
<td>Allows you to delete any accelerator key bound to the menu element.</td>
</tr>
<tr>
<td>Disabled</td>
<td>Allows you to disable a menu element.</td>
</tr>
<tr>
<td>Toggle-Box</td>
<td>Allows you to define the menu element as a toggle box.</td>
</tr>
<tr>
<td>Action</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>Insert</td>
<td>Allows you to insert a new menu element.</td>
</tr>
<tr>
<td>Delete</td>
<td>Allows you to delete the selected menu element.</td>
</tr>
<tr>
<td>Rule</td>
<td>Allows you to create a rule line in the pull-down or pop-up menu (menu-items only).</td>
</tr>
<tr>
<td>Skip</td>
<td>Allows you to add blank lines to the pull-down or pop-up menu (menu-items only).</td>
</tr>
<tr>
<td>Up</td>
<td>Allows you to move the selected menu element one position up.</td>
</tr>
<tr>
<td>Down</td>
<td>Allows you to move the selected menu element one position down.</td>
</tr>
<tr>
<td>&lt;&lt;</td>
<td>Allows you to remove a dash from left side of the selected menu element. This dash indicates the menu element position in the menu hierarchy.</td>
</tr>
<tr>
<td>&gt;&gt;</td>
<td>Allows you to add a dash to the left side of the current menu element. This dash indicates the menu element position in the menu hierarchy.</td>
</tr>
<tr>
<td>New Menu Element</td>
<td>Allows you to create a new menu element at the same level as the selected menu element.</td>
</tr>
</tbody>
</table>

For more information, see Property Sheet dialogs on page 702

**Property Sheet for Browse**

The Property Sheet - Browse dialog allows you to modify the browse attributes.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Enter a new name for the browse.</td>
</tr>
<tr>
<td>ToolTip</td>
<td>Allows you to enter text for tool tip help.</td>
</tr>
<tr>
<td>Help</td>
<td>Enter a Help Identifier associated with this field. This field is used only in .HLP help files to support What's This? context-sensitive help.</td>
</tr>
<tr>
<td>Widget ID</td>
<td>Enter a widget ID for the browse.</td>
</tr>
<tr>
<td>Title</td>
<td>Enter a new title for the browse.</td>
</tr>
<tr>
<td>Query</td>
<td>Displays the database query for the browse. You can edit the query by clicking Query, which opens the Query Builder dialog.</td>
</tr>
<tr>
<td>Fields</td>
<td>Displays the Column Editor dialog which allows you to add fields to the database query for the browse.</td>
</tr>
<tr>
<td>Locked Columns</td>
<td>Enter a number of columns to lock in the browse view. As you scroll horizontally, these locked columns remain freeze in the view, with the remaining columns scrolling horizontally.</td>
</tr>
<tr>
<td>Max Data Guess</td>
<td>Enter an estimate for the number of records in the browse query. The browse uses this number to visually depict query location (set the thumb position) in the vertical scroll.</td>
</tr>
</tbody>
</table>
### Column
Enter a window column value (in character units) where you want to place the browse upper left or right corner. This depends upon the position setting. For example, if Left-Align is chosen, the value in Column refers to the browse upper left corner.

### Row
Enter a window row value (in character units) where you want to place the browse upper left or right corner. This depends upon the position setting. For example, if Left-Align is chosen, the value in Row refers to the browse upper left corner.

### Width
Enter a value (in character units) to specify the width of the browse.

### Height
Enter a value (in character units) to specify the height of the browse.

### Left-Align
Select this option to position the browse left edge along the column position specified.

### Right-Align
Select this option to position the browse right edge along the column position specified.

### Other Settings
This section allows you to select the run-time attributes and other ABL related options or accept the defaults.

- **Color**
  Displays the Choose Color dialog which allows you change browse background and foreground colors.

- **Font**
  Displays the Choose Font dialog which allows you to change the browse font at run-time.

- **Popup Menu**
  Displays the Property Sheet - Popup Menu dialog which allows you to create a pop-up menu for the browse.

- **Translation Attributes**
  Displays the String Attributes dialog which allows you to enter translation attributes for the browse.

- **Advanced**
  Displays the Advanced Properties dialog which allows you to edit advanced properties for the selected browse.

For more information, see Property Sheet dialogs on page 702

### Property Sheet for Button

The Property Sheet - Button dialog allows you to modify the button properties.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Enter a new name for the button or accept the default.</td>
</tr>
<tr>
<td>Label</td>
<td>Enter text for the button label. The GUI Designer displays the text on the button unless you define an up image.</td>
</tr>
<tr>
<td>ToolTip</td>
<td>Enter text for tool tip help.</td>
</tr>
<tr>
<td><strong>Help</strong></td>
<td>Enter a Help Identifier associated with this field. This field is used only in .HLP help files to support What's This? context-sensitive help.</td>
</tr>
</tbody>
</table>
| **Up** | Displays the Choose Image dialog which allows you to select an image for the button in its up state (image displayed when the button is not depressed with the mouse).  
**Note:** The GUI Designer adds the selected image as the IMAGE-UP option to the DEFINE BUTTON statement. |
| **Down** | Displays the Choose Image dialog which allows you to select an image for the button in its down state (image displayed when the button is depressed with the mouse). A down image has no effect unless you have defined an up image.  
**Note:** The GUI Designer adds the selected image as the IMAGE-DOWN option to the DEFINE BUTTON statement. |
| **Insensitive** | Displays the Choose Image dialog which allows you to select an image for the button in its insensitive (disabled) state. An insensitive image has no effect unless you have defined an up image.  
**Note:** The GUI Designer adds the selected image as the IMAGE-INSENSITIVE option to the DEFINE BUTTON statement. |
| **Column** | Enter a window column value (in character units) where you want to place the button upper left or right corner.  
This depends upon the position setting. For example, if Left-Align is chosen, the value in Column refers to the button upper left corner. |
| **Row** | Enter a window row value (in character units) where you want to place the button upper left or right corner.  
This depends upon the position setting. For example, if Left-Align is chosen, the value in Row refers to the button upper left corner. |
| **Width** | Enter a value (in character units) to specify the width of the button. |
| **Height** | Enter a value (in character units) to specify the height of the button. |
| **Left-Align** | Select this option to position the button left edge along the column position specified. |
| **Right-Align** | Select this option to position the button right edge along the column position specified. |
| **Other Settings** | This section allows you to select the run-time attributes and other ABL related options or accept the defaults. |
| **Color** | Displays the Choose Color dialog which allows you to change background and foreground colors on the button. |
| **Font** | Displays the Choose Font dialog which allows you to change the font of the button text. |
| **Popup Menu** | Displays the Property Sheet - Popup Menu dialog which allows you to create a pop-up menu for the button. |
| **Translation Attributes** | Displays the String Attributes dialog which allows you to enter translation attributes for the button. |
| **Advanced** | Displays the Advanced Properties dialog which allows you to edit more attributes for the button. |

For more information, see Property Sheet dialogs on page 702

**Property Sheet for Combo Box**

The Property Sheet - Combo Box dialog allows you to modify the combo box properties.

This dialog includes the following options:

| **Object** | Enter a new name for the combo box or accept the default. |
| **Note:** If you make the combo box a database field, the combo box uses the database field name. |
| **Tooltip** | Allows you to enter text for tool tip help. |
| **Help** | Enter a Help Identifier associated with this field. This field is used only in .HLP help files to support What's This? context-sensitive help. |
| **Label** | Enter a new label for the combo box or accept the default. |
| **Note:** If you make the combo box a database field, the combo box uses the database field label. |

If you want to suppress the combo box label, select the No-Label check box. GUI Designer adds the No-Label option to the Frame phrase.

| **List-Items** | Enter items to display in the list component of the combo box. |
| **Inner Lines** | Enter the number of items visible at one time in the list component of the combo box. |
| **Define As** | Select a data type for the combo box. |
| **Format** | Enter a display format for the combo box. |

You can enter a display format for the combo box by clicking Format that opens Format Assistant dialog.
**Column**
Enter a window column value (in character units) where you want to place the combo box upper left or right corner.

This depends upon the position setting and whether the combo box has a label. For example, if Left-Align is chosen, the value in Column refers to the combo box upper left corner.

**Row**
Enter a window row value (in character units) where you want to place the combo box upper left or right corner.

This depends upon the position setting and whether the combo box has a label. For example, if Left-Align is chosen, the value in Row refers to the combo box upper left corner.

**Width**
Enter a value (in character units) to specify the width of the combo box.

**Height**
Enter a value (in character units) to specify the height of the combo box.

**Left-Align**
Select this option to position the combo box left edge (or label left edge) along the column position specified.

**Colon-Align**
Select this option to position the combo box label colon (:) along the column position specified.

**Note:** If there is no label, then the GUI Designer aligns the combo box as if it had a label.

**Right-Align**
Select this option to position the combo box right edge along the column position specified.

**Other Settings**
This section allows you to select run-time attributes and other ABL related options or accept the defaults.

- **Color**
  Displays the Choose Color dialog which allows you to change combo box background and foreground colors.

- **Font**
  Displays the Choose Font dialog which allows you to change the combo box font (excluding combo box label).

- **Popup Menu**
  Displays the Property Sheet - Popup Menu dialog which allows you to create a pop-up menu for the combo box.

- **Translation Attributes**
  Displays the String Attributes dialog which allows you to enter translation attributes for the combo box label, title, and display format.

- **Database Field**
  Displays the Field Selector dialog which allows you to connect a database field to the combo box.

- **Advanced**
  Displays the Advanced Properties dialog which allows you to edit advanced properties for the combo box.

For more information, see Property Sheet dialogs on page 702
## Property Sheet for Editor

The **Property Sheet - Editor** dialog allows you to modify the editor properties. You can select Character or LongChar as a data type for an editor.

This dialog includes the following options:

<table>
<thead>
<tr>
<th><strong>Object</strong></th>
<th>Enter a new name for the editor or accept the default.</th>
</tr>
</thead>
</table>
| **Define As**  | Select a data type for the editor from the list. You can select either Character or LongChar.  
|                | You must make an editor LongChar to display a large character object (CLOB). |
| **Note**: When you select **LongChar**, the **Large** option in the **Other Settings** section is selected, and will be disabled to modify. |
| **Maximum Characters** | Enter a value to specify the maximum number of characters that can be contained in the editor. |
| **Tooltip**    | Enter text for tool tip help. |
| **Help**       | Enter a Help Identifier associated with this field. This field is used only in .HLP help files to support What's This? context-sensitive help. |
| **Widget ID**  | Enter a widget ID for the editor. |
| **Column**     | Enter a window column value (in character units) where you want to place the editor upper left or right corner.  
|                | This depends upon the position setting. For example, if Left-Align is chosen, the value in Column refers to the radio set upper left corner. |
| **Row**        | Enter a window row value (in character units) where you want to place the editor upper left or right corner.  
|                | This depends upon the position setting. For example, if Left-Align is chosen, the value in Row refers to the radio set upper left corner. |
| **Width**      | Enter a value (in character units) to specify the width of the editor. |
| **Height**     | Enter a value (in character units) to specify the height of the editor. |
| **Left-Align** | Select this option to position the editor left edge along the column position specified. |
| **Right-Align**| Select this option to position the editor right edge along the column position specified. |
| **Other Settings** | This section allows you to select run-time attributes and other ABL related options or accept the defaults. |
| **Color**      | Displays the **Choose Color** dialog which allows you to change editor background and foreground colors. |
### Property Sheet for Fill In

The **Property Sheet - Fill In** dialog allows you to modify the fill-in properties. This dialog includes the following options:

<table>
<thead>
<tr>
<th><strong>Object</strong></th>
<th>Enter a new name for the fill-in or accept the default.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>If you make the fill-in a database field, the fill-in uses the database field name.</td>
</tr>
<tr>
<td><strong>Label</strong></td>
<td>Enter a new label for the fill-in or accept the default.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>If you make the fill-in a database field, the fill-in uses the database field label.</td>
</tr>
<tr>
<td></td>
<td>If you want to suppress the fill-in label, select the <strong>No-Label</strong> check box. GUI Designer adds the No-Label option to the Frame phrase.</td>
</tr>
<tr>
<td><strong>Define As</strong></td>
<td>Select a data type for the fill-in from the list.</td>
</tr>
<tr>
<td><strong>Format</strong></td>
<td>Enter a display format for the fill-in. You can enter a display format for the combo box by clicking <strong>Format</strong> that opens Format Assistant dialog.</td>
</tr>
<tr>
<td><strong>Tooltip</strong></td>
<td>Allows you to enter text for tool tip help.</td>
</tr>
<tr>
<td><strong>Help ID</strong></td>
<td>Enter a Help Identifier associated with this field. This field is used only in .HLP help files to support What's This? context-sensitive help.</td>
</tr>
<tr>
<td><strong>Column</strong></td>
<td>Enter a window column value (in character units) where you want to place the fill-in upper left or right corner. This depends upon the position setting and whether the fill-in has a label. For example, if Left-Align is chosen, the value in Column refers to the fill-in upper left corner.</td>
</tr>
</tbody>
</table>
### Property Sheet for Frame

The Property Sheet - Frame dialog allows you to modify the frame properties.

This dialog includes the following options:

<table>
<thead>
<tr>
<th><strong>Object</strong></th>
<th><strong>Title</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter a new name for the frame.</td>
<td>Enter a new title for the frame.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Query</td>
<td>Displays the database query for the frame. You can edit the query by clicking Query, which opens the Query Builder dialog.</td>
</tr>
<tr>
<td>Column</td>
<td>Enter a value (in character units) where you want to place the frame upper left corner relative to the design window.</td>
</tr>
<tr>
<td>Row</td>
<td>Enter a value (in character units) where you want to place the frame upper left corner relative to the design window.</td>
</tr>
<tr>
<td>Width</td>
<td>Enter a value (in character units) to specify the width of the frame.</td>
</tr>
<tr>
<td>Height</td>
<td>Enter a value (in character units) to specify the height of the frame.</td>
</tr>
<tr>
<td>Virtual Width</td>
<td>Enter the maximum and virtual width (in character units) of the frame.</td>
</tr>
<tr>
<td>Note:</td>
<td>This field is enabled when you select the Scrollable check box in the Other Settings section.</td>
</tr>
<tr>
<td>Virtual Height</td>
<td>Enter the maximum and virtual height (in character units) of the frame.</td>
</tr>
<tr>
<td>Note:</td>
<td>This field is enabled when you select the Scrollable check box in the Other Settings section.</td>
</tr>
<tr>
<td>Other Settings</td>
<td>This section allows you to select the run-time attributes and other ABL related options or accept the defaults.</td>
</tr>
<tr>
<td>Color</td>
<td>Displays the Choose Color dialog which allows you to change frame background color and foreground color for field-level objects in the frame.</td>
</tr>
<tr>
<td>Font</td>
<td>Displays the Choose Font dialog which allows you to change the font for field-level objects in the frame.</td>
</tr>
<tr>
<td>Popup Menu</td>
<td>Displays the Property Sheet - Popup Menu dialog which allows you to create a pop-up menu for the frame.</td>
</tr>
<tr>
<td>Translation Attributes</td>
<td>Displays the String Attributes dialog which allows you to enter translation attributes for the frame title.</td>
</tr>
<tr>
<td>Tab Order</td>
<td>Displays the Tab Editor dialog which allows you to change the tab order of field-level objects in the frame.</td>
</tr>
<tr>
<td>Advanced</td>
<td>Displays the Advanced Properties dialog which allows you to edit advanced properties for the selected frame.</td>
</tr>
</tbody>
</table>

For more information, see Property Sheet dialogs on page 702.
Property Sheet for Image

The Property Sheet - Image dialog allows you to modify the image properties.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Enter a name for the image.</td>
</tr>
<tr>
<td>Image</td>
<td>Displays the Choose Image dialog which allows you to select an image (.ico file) for the image.</td>
</tr>
<tr>
<td>Tooltip</td>
<td>Allows you to enter text for tool tip help.</td>
</tr>
<tr>
<td>Widget ID</td>
<td>Allows you to enter a widget ID for the image.</td>
</tr>
<tr>
<td>Column</td>
<td>Enter a window column value (in character units) where you want to place the image upper left corner relative to the design canvas.</td>
</tr>
<tr>
<td>Row</td>
<td>Enter a window row value (in character units) where you want to place the image upper left corner relative to the design canvas.</td>
</tr>
<tr>
<td>Width</td>
<td>Allows you to enter a value in character units to specify the width of the image.</td>
</tr>
<tr>
<td>Height</td>
<td>Allows you to enter a value in character units to specify the height of the image.</td>
</tr>
<tr>
<td>Left-Align</td>
<td>Select this option to position the image left edge along the column position specified.</td>
</tr>
<tr>
<td>Right-Align</td>
<td>Select this option to position the image right edge along the column position specified.</td>
</tr>
<tr>
<td>Other Settings</td>
<td>This section allows you to select run-time attributes and other ABL related options or accept the defaults.</td>
</tr>
<tr>
<td>Translation Attributes</td>
<td>Displays the String Attributes dialog which allows you to enter translation attributes for the image.</td>
</tr>
<tr>
<td>Advanced</td>
<td>Displays the Advanced Properties dialog which allows you to edit advanced properties for the selected image.</td>
</tr>
</tbody>
</table>

For more information, see Property Sheet dialogs on page 702

Property Sheet for Radio Set

The Property Sheet - Radio Set dialog allows you to modify the radio set attributes.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Enter a new name for the radio set or accept the default.</td>
</tr>
<tr>
<td>Buttons</td>
<td>Enter the label for each radio button or accept the default. Radio buttons are a comma separated list. For example, &quot;item-string&quot;, value, &quot;item-string&quot;, value. Item strings must be enclosed within quotes and must be appropriate to the data type. The GUI Designer adds these values to the RADIO-SET phrase.</td>
</tr>
<tr>
<td><strong>Define As</strong></td>
<td>Select a data type for the radio set from the list.</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td><strong>Tooltip</strong></td>
<td>Enter text for tool tip help.</td>
</tr>
<tr>
<td><strong>Help</strong></td>
<td>Enter a Help Identifier associated with this field. This field is used only in .HLP help files to support What's This? context-sensitive help.</td>
</tr>
<tr>
<td><strong>Widget ID</strong></td>
<td>Enter a widget ID for the radio set.</td>
</tr>
<tr>
<td><strong>Column</strong></td>
<td>Enter a window column value (in character units) where you want to place the radio set upper left or right corner. This depends upon the position setting. For example, if Left-Align is chosen, the value in Column refers to the radio set upper left corner.</td>
</tr>
<tr>
<td><strong>Row</strong></td>
<td>Enter a window row value (in character units) where you want to place the radio set upper left or right corner. This depends upon the position setting. For example, if Left-Align is chosen, the value in Row refers to the radio set upper left corner.</td>
</tr>
<tr>
<td><strong>Width</strong></td>
<td>Enter a value (in character units) to specify the width of the radio set.</td>
</tr>
<tr>
<td><strong>Height</strong></td>
<td>Enter a value (in character units) to specify the height of the radio set.</td>
</tr>
<tr>
<td><strong>Left-Align</strong></td>
<td>Select this option to position the radio set left edge along the column position specified.</td>
</tr>
<tr>
<td><strong>Right-Align</strong></td>
<td>Select this option to position the radio set right edge along the column position specified.</td>
</tr>
<tr>
<td><strong>Other Settings</strong></td>
<td>This section allows you to select run-time attributes and other ABL related options or accept the defaults.</td>
</tr>
<tr>
<td><strong>Color</strong></td>
<td>Displays the Choose Color dialog which allows you to change radio set background and foreground colors.</td>
</tr>
<tr>
<td><strong>Font</strong></td>
<td>Displays the Choose Font dialog which allows you to change the radio set font (excluding radio set label).</td>
</tr>
<tr>
<td><strong>Popup Menu</strong></td>
<td>Displays the Property Sheet - Popup Menu dialog which allows you to create a pop-up menu for the radio set.</td>
</tr>
<tr>
<td><strong>Translation Attributes</strong></td>
<td>Displays the String Attributes dialog which allows you to enter translation attributes for radio set help.</td>
</tr>
<tr>
<td><strong>Database Field</strong></td>
<td>Displays the Field Selector dialog which allows you to connect a database field to the radio set.</td>
</tr>
<tr>
<td>Advanced</td>
<td>Displays the Advanced Properties dialog which allows you to edit more attributes for the radio set.</td>
</tr>
</tbody>
</table>

For more information, see Property Sheet dialogs on page 702
**Property Sheet for Rectangle**

The **Property Sheet - Rectangle** dialog allows you to modify the rectangle properties.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Enter a name for the rectangle widget.</td>
</tr>
<tr>
<td>Edge Pixels</td>
<td>Allows you to enter the width of the rectangle border (in pixels). The rectangle foreground color is used for its edge.</td>
</tr>
<tr>
<td>Tooltip</td>
<td>Allows you to enter text for tool tip help.</td>
</tr>
<tr>
<td>Widget ID</td>
<td>Allows you to enter a widget ID for the rectangle.</td>
</tr>
<tr>
<td>Column</td>
<td>Enter a window column value (in character units) where you want to place the rectangle upper left or right corner. This depends upon the position setting. For example, if Left-Align is chosen, the value in Column refers to the rectangle upper left corner.</td>
</tr>
<tr>
<td>Row</td>
<td>Enter a window row value (in character units) where you want to place the rectangle upper left or right corner. This depends upon the position setting. For example, if Left-Align is chosen, the value in Row refers to the rectangle upper left corner.</td>
</tr>
<tr>
<td>Width</td>
<td>Allows you to enter a value in character units to specify the width of the rectangle.</td>
</tr>
<tr>
<td>Height</td>
<td>Allows you to enter a value in character units to specify the height of the rectangle.</td>
</tr>
<tr>
<td>Left-Align</td>
<td>Select this option to position the rectangle left edge along the column position specified.</td>
</tr>
<tr>
<td>Right-Align</td>
<td>Select this option to position the rectangle right edge along the column position specified.</td>
</tr>
<tr>
<td>Other Settings</td>
<td>This section allows you to select run-time attributes and other ABL related options or accept the defaults.</td>
</tr>
<tr>
<td>Color</td>
<td>Displays the Choose Color dialog which allows you to change rectangle background color and foreground color.</td>
</tr>
<tr>
<td>Translation Attributes</td>
<td>Displays the String Attributes dialog which allows you to enter translation attributes for the rectangle title.</td>
</tr>
<tr>
<td>Advanced</td>
<td>Displays the Advanced properties for Rectangle dialog which allows you to edit advanced properties for the selected rectangle.</td>
</tr>
</tbody>
</table>

**Note:** When the Filled check box is not selected in the Other Settings section, the rectangle will have no background color.

For more information, see Property Sheet dialogs on page 702.
### Property Sheet for Selection List

The **Property Sheet - Selection List** dialog allows you to modify the selection list properties. This dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object</strong></td>
<td>Enter a new name for the selection list or accept the default.</td>
</tr>
<tr>
<td><strong>Tooltip</strong></td>
<td>Enter text for tool tip help.</td>
</tr>
<tr>
<td><strong>Help</strong></td>
<td>Enter a Help Identifier associated with this field. This field is used only in .HLP help files to support What's This? context-sensitive help.</td>
</tr>
<tr>
<td><strong>List Items</strong></td>
<td>Enter items to display in the selection list.</td>
</tr>
<tr>
<td><strong>Column</strong></td>
<td>Enter a window column value (in character units) where you want to place the selection list upper left or right corner.</td>
</tr>
<tr>
<td></td>
<td>This depends upon the position setting. For example, if Left-Align is chosen, the value in Column refers to the selection list upper left corner.</td>
</tr>
<tr>
<td><strong>Row</strong></td>
<td>Enter a window row value (in character units) where you want to place the selection list upper left or right corner.</td>
</tr>
<tr>
<td></td>
<td>This depends upon the position setting. For example, if Left-Align is chosen, the value in Row refers to the selection list upper left corner.</td>
</tr>
<tr>
<td><strong>Width</strong></td>
<td>Enter a value (in character units) to specify the width of the selection list.</td>
</tr>
<tr>
<td><strong>Height</strong></td>
<td>Enter a value (in character units) to specify the height of the selection list.</td>
</tr>
<tr>
<td><strong>Left-Align</strong></td>
<td>Select this option to position the selection list left edge along the column position specified.</td>
</tr>
<tr>
<td><strong>Right-Align</strong></td>
<td>Select this option to position the selection list right edge along the column position specified.</td>
</tr>
<tr>
<td><strong>Other Settings</strong></td>
<td>This section allows you to select run-time attributes and other ABL related options or accept the defaults.</td>
</tr>
<tr>
<td><strong>Color</strong></td>
<td>Displays the Choose Color dialog which allows you to change selection list background and foreground colors.</td>
</tr>
<tr>
<td><strong>Fonts</strong></td>
<td>Displays the Choose Font dialog which allows you to change the selection list font.</td>
</tr>
<tr>
<td><strong>Popup Menu</strong></td>
<td>Displays the Property Sheet - Popup menu dialog which allows you to create a pop-up menu for the selection list.</td>
</tr>
<tr>
<td>Translation Attributes</td>
<td>Displays the <strong>String Attributes</strong> dialog which allows you to enter translation attributes for selection list help.</td>
</tr>
<tr>
<td>------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Database Field</td>
<td>Displays the <strong>Field Selector</strong> dialog which allows you to connect a database field to the selection list.</td>
</tr>
<tr>
<td>Advanced</td>
<td>Displays the <strong>Advanced Properties</strong> dialog which allows you to edit more attributes for the selection list.</td>
</tr>
</tbody>
</table>

For more information, see **Property Sheet dialogs** on page 702

**Property Sheet for Slider**

The **Property Sheet - Slider** dialog allows you to modify the slider properties.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Object</th>
<th>Enter a name for the toggle box or accept the default.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min Value</td>
<td>Enter a minimum integer value for the slider. The minimum value cannot be less than -32,768.</td>
</tr>
<tr>
<td>Max Value</td>
<td>Enter a maximum integer value for the slider. The maximum value cannot exceed 32,767.</td>
</tr>
</tbody>
</table>
| Tic Marks  | Select to display tick marks on the slider, and select the position of the marks: Top/Left, Bottom/Right, or Both.  
            | The default option is None, which indicates no marks are displayed on the slider. |
| Frequency  | Enter the value frequency (in integer) for each tick mark.  
            | **Note**: This option is enabled when any value except "None" is selected from the Tic Marks list. |
| ToolTip    | Enter text for the tool tip help. |
| Help       | Enter a Help Identifier associated with this field. This field is used only in .HLP help files to support What's This? context-sensitive help. |
| Widget ID  | Enter a widget ID for the toggle box. |
| Column     | Enter a window column value (in character units) where you want to place the toggle box upper left or right corner.  
            | This depends upon the position setting. For example, if Left-Align is chosen, the value in Column refers to the toggle box upper left corner. |
| Row        | Enter a window row value (in character units) where you want to place the toggle box upper left or right corner.  
            | This depends upon the position setting. For example, if Left-Align is chosen, the value in Row refers to the toggle box upper left corner. |
### Property Sheet for Text

The **Property Sheet - Text** dialog allows you to modify the text properties. This dialog includes the following options:

<table>
<thead>
<tr>
<th>Text</th>
<th>Enter text you want to display for the selected text widget.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ToolTip</td>
<td>Allows you to enter text for tool tip help.</td>
</tr>
<tr>
<td>Column</td>
<td>Enter a window column value (in character units) where you want to place the text object upper left or right corner. This depends upon the position setting. For example, if Left-Align is chosen, the value in Column refers to the text string upper left corner.</td>
</tr>
</tbody>
</table>

For more information, see [Property Sheet dialogs](#) on page 702
Enter a window row value (in character units) where you want to place the text object upper left or right corner. This depends upon the position setting. For example, if Left-Align is chosen, the value in Row refers to the text string upper left corner.

Enter a value (in character units) to specify the width of the text object.

Enter a value (in character units) to specify the height of the text object.

Select this option to position the text object left edge along the column position specified.

Select this option to position the text object right edge along the column position specified.

Displays the **Choose Color** dialog which allows you to change text object background and foreground colors.

Displays the **Choose Font** dialog which allows you to change the text object font.

Displays the **String Attributes** dialog which allows you to enter translation attributes for the selected text widget.

Displays the **Advanced Properties** dialog which allows you to edit advanced properties for the selected text widget.

For more information, see Property Sheet dialogs on page 702

**Property Sheet for Toggle Box**

The **Property Sheet - Toggle Box** dialog allows you to modify the toggle box properties. This dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Enter a name for the toggle box or accept the default.</td>
</tr>
<tr>
<td>Label</td>
<td>Enter a label for the toggle box.</td>
</tr>
<tr>
<td>ToolTip</td>
<td>Enter text for tool tip help.</td>
</tr>
<tr>
<td>Help</td>
<td>Enter a Help Identifier associated with this field. This field is used only in .HLP help files to support What's This? context-sensitive help.</td>
</tr>
<tr>
<td>Widget ID</td>
<td>Enter a widget ID for the toggle box.</td>
</tr>
<tr>
<td>Column</td>
<td>Enter a window column value (in character units) where you want to place the toggle box upper left or right corner. This depends upon the position setting. For example, if Left-Align is chosen, the value in Column refers to the toggle box upper left corner.</td>
</tr>
<tr>
<td><strong>Row</strong></td>
<td>Enter a window row value (in character units) where you want to place the toggle box upper left or right corner. This depends upon the position setting. For example, if Left-Align is chosen, the value in Row refers to the toggle box upper left corner.</td>
</tr>
<tr>
<td><strong>Width</strong></td>
<td>Enter a value (in character units) to specify the width of the toggle box.</td>
</tr>
<tr>
<td><strong>Height</strong></td>
<td>Enter a value (in character units) to specify the height of the toggle box.</td>
</tr>
<tr>
<td><strong>Left-Align</strong></td>
<td>Select this option to position the toggle box left edge along the column position specified.</td>
</tr>
<tr>
<td><strong>Right-Align</strong></td>
<td>Select this option to position the toggle box right edge (or label right edge) along the column position specified.</td>
</tr>
<tr>
<td><strong>Other Settings</strong></td>
<td>This section allows you to select run-time attributes and other ABL related options, or accept the defaults.</td>
</tr>
<tr>
<td><strong>Color</strong></td>
<td>Displays the Choose Color dialog which allows you to change toggle box background and foreground colors.</td>
</tr>
<tr>
<td><strong>Font</strong></td>
<td>Displays the Choose Font dialog which allows you to change the toggle box font (excluding toggle box label).</td>
</tr>
<tr>
<td><strong>Popup Menu</strong></td>
<td>Displays the Property Sheet - Popup Menu dialog which allows you to create a pop-up menu for the toggle box.</td>
</tr>
<tr>
<td><strong>Translation Attributes</strong></td>
<td>Displays the String Attributes dialog which allows you to enter translation attributes for the toggle box label and help.</td>
</tr>
<tr>
<td><strong>Database Field</strong></td>
<td>Displays the Field Selector dialog which allows you to connect a database field to the toggle box.</td>
</tr>
<tr>
<td><strong>Advanced</strong></td>
<td>Displays the Advanced Properties dialog which allows you to edit more attributes for the toggle box.</td>
</tr>
</tbody>
</table>

For more information, see Property Sheet dialogs on page 702

**Property Sheet for OCX**

The Property Sheet - OCX dialog allows you to modify the OCX control frame properties.

This dialog includes the following options:

<p>| <strong>Object</strong> | Enter a new name for the OCX control frame or accept the default. |</p>
<table>
<thead>
<tr>
<th>Help ID</th>
<th>Enter a Help Identifier associated with this field. This field is used only in .HLP help files to support What's This? context-sensitive help.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Widget ID</td>
<td>Enter a widget ID for the OCX control frame.</td>
</tr>
<tr>
<td>Column</td>
<td>Enter a window column value (in character units) where you want to place the control frame upper left corner.</td>
</tr>
<tr>
<td>Row</td>
<td>Enter a window row value (in character units) where you want to place the OCX control frame upper left corner.</td>
</tr>
<tr>
<td>Width</td>
<td>Enter a value (in character units) to specify the width of the OCX control frame.</td>
</tr>
<tr>
<td>Height</td>
<td>Enter a value (in character units) to specify the height of the OCX control frame.</td>
</tr>
<tr>
<td>Other Settings</td>
<td>This section allows you to select the run-time attributes and other ABL related options or accept the defaults.</td>
</tr>
<tr>
<td>Advanced</td>
<td>Displays the Advanced Properties dialog which allows you to edit advanced properties for the OCX control frame.</td>
</tr>
</tbody>
</table>

For more information, see Property Sheet dialogs on page 702

**Property Sheet for SmartObjects**

The Property Sheet - SmartObjects dialog allows you to modify the properties for a SmartObject instance. This property sheet includes the following options:

<table>
<thead>
<tr>
<th>Object</th>
<th>Enter a new name for the SmartObject instance or accept the default.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Displays the SmartObject type for the selected SmartObject instance.</td>
</tr>
<tr>
<td>Master File</td>
<td>Displays the name of the master file from which the selected SmartObject instance is created.</td>
</tr>
<tr>
<td>File</td>
<td>Displays the Modify Master File Name dialog which allows you to change the name of the master file from which the selected SmartObject instance is created.</td>
</tr>
<tr>
<td>Instance Properties</td>
<td>Displays the instance properties for the selected SmartObject instance.</td>
</tr>
<tr>
<td>Edit</td>
<td>Allows you to view the available properties and their corresponding values for the selected SmartObject. This option displays the SmartObject Properties dialog.</td>
</tr>
<tr>
<td>Parameterize as Variable</td>
<td>Activate to use the selected SmartObject instance as a placeholder. If you check this box, you must also add code to load the proper object at run time.</td>
</tr>
<tr>
<td>Variable</td>
<td>Select a variable name to use when parameterizing the selected SmartObject instance. This field is only displayed when Parameterize as Variable is selected.</td>
</tr>
</tbody>
</table>
Open the **SmartLinks** dialog that enables you to view, add, and remove SmartLinks between SmartObject instances in the current design window.

Open the **SmartInfo** dialog that enables you to view information on the SmartLinks supported by the selected SmartObject instance, and the database tables used and supplied the SmartObject instance.

For more information, see Property Sheet dialogs on page 702

**Other Settings section of Property Sheet dialog**

You can activate various toggles in the **Other Settings** section of an object Property Sheet to govern the object’s run-time characteristics. Below is an alphabetical compilation of object run-time attributes and other object property options displayed as toggle boxes in the **Other Settings** section.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-D</td>
<td>Activate for objects (including dialog boxes and frames) to have a uniform three-dimensional look. The AppBuilder sets the 3-D attribute to TRUE. (Windows only)</td>
</tr>
<tr>
<td>Auto-End-Key</td>
<td>Activate to make the button act like ENDKEY when chosen. The AppBuilder sets the AUTO-END-KEY attribute for the button to TRUE.</td>
</tr>
<tr>
<td>Auto-Go</td>
<td>Activate to make the button act like GO when chosen. The AppBuilder sets the AUTO-GO attribute for the button to TRUE.</td>
</tr>
<tr>
<td>Auto-Indent</td>
<td>Activate to make each new line of text typed in the editor object automatically indent to line up with the preceding line. The AppBuilder sets the AUTO-INDENT attribute for the editor object to TRUE.</td>
</tr>
<tr>
<td>Auto-Resize</td>
<td>Activate to make the object automatically resize when its contents are changed. The AppBuilder sets the AUTO-RESIZE attribute for the object to TRUE.</td>
</tr>
<tr>
<td>Auto-Return</td>
<td>Activate to set the AUTO-RETURN attribute for the fill-in to TRUE. When the last character is typed in a field, users automatically leave the field as if they had pressed RETURN.</td>
</tr>
<tr>
<td>Blank</td>
<td>Activate to set the BLANK attribute for the fill-in or text object to TRUE. Any current value or characters typed in a fill-in are not echoed to the screen.</td>
</tr>
<tr>
<td>Cancel Button</td>
<td>Activate to specify the button as the one chosen for all ESC key label events for the frame or dialog box. The AppBuilder adds the CANCEL-BUTTON attribute to the frame or dialog box. Choosing this attribute supersedes any existing cancel button defined for the frame.</td>
</tr>
<tr>
<td>Column-Scrolling</td>
<td>Activate to enable horizontal scrolling of the browse object by column at run time. The AppBuilder sets the COLUMN-SCROLLING attribute for the browse object to TRUE. (Deactivate to enable horizontal scrolling by character.) (Windows only)</td>
</tr>
<tr>
<td>Control-Box</td>
<td>Activate to enable the maximize and minimize buttons at the right end of the Window's title bar and the close button at the left end. (Windows only)</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Convert-3D-Colors</td>
<td>Activate to convert image colors to corresponding platform-specific system colors. This often produces an unwelcome effect. (Windows only)</td>
</tr>
<tr>
<td>Deblank</td>
<td>Activate to remove leading blanks from CHARACTER fill-ins enabled for input after a change to its SCREEN-VALUE attribute during input. The AppBuilder sets DEBLANK attribute for the object to TRUE.</td>
</tr>
<tr>
<td>Default-Button</td>
<td>Activate to specify the button as the one chosen for all RETURN key function events, whether or not it has focus (unless there is another object in the frame or dialog box that normally takes a RETURN key function event). The AppBuilder adds the DEFAULT-BUTTON attribute to the frame or dialog box. Choosing this attribute supersedes any existing default button defined for the frame.</td>
</tr>
<tr>
<td>Default-Style</td>
<td>Activate for the AppBuilder to add an extra margin around the button to make it a default style button (default button margins are platform-specific). The AppBuilder adds the DEFAULT option to the DEFINE BUTTON statement.</td>
</tr>
<tr>
<td>Disable-Auto-Zap</td>
<td>Activate to prevent the Fill-in from automatically clearing its input field whenever it gets the focus.</td>
</tr>
<tr>
<td>Display</td>
<td>Activate to DISPLAY the variable value in the default enabling procedure. (You do not need to activate Display to make the object visible. Display causes the enabling procedure to populate the object with the variable data value.)</td>
</tr>
<tr>
<td>Down</td>
<td>Activate for the AppBuilder to allow multiple iterations of the frame if there's room in the frame.</td>
</tr>
<tr>
<td>Drag-Enabled</td>
<td>Activate to set the DRAG-ENABLED attribute for the selection list to TRUE. The user can simultaneously hold down the mouse select button and drag the mouse through the list. As the drag passes over an item, the item is highlighted. When the user releases the select button, the highlighted item becomes the selected item.</td>
</tr>
<tr>
<td>Drop-Target</td>
<td>Activate to generate an event when some object is dragged and dropped onto a window. You can write code to intercept and respond to such events under program control.</td>
</tr>
<tr>
<td>Enable</td>
<td>Activate to enable the field-level object for input. The AppBuilder includes the object in the ENABLE list in the default enabling procedure (enable_UI).</td>
</tr>
<tr>
<td>Expansion</td>
<td>Activate to specify each button in a radio set to fill the radio set bounding box. The AppBuilder sets the EXPAND attribute to TRUE. (Horizontal must be activated for Expand to apply.)</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Expandable</td>
<td>Activate to set a browse's EXPANDABLE attribute to TRUE, the right-most browse-column is extended horizontally to the browse's right edge, if necessary, to cover any white space that might appear there. The right-most browse-column expands only when there is no horizontal scroll bar.</td>
</tr>
<tr>
<td>Explicit-Position</td>
<td>Activate to position the window or dialog box at the screen location indicated by the values in the Column and Row fields at run time. The AppBuilder adds the COLUMN and ROW attributes to the CREATE WIDGET statement. (When deactivated the AppBuilder removes the Column and Row attributes: windows are placed by the program manager. Dialog boxes are centered over their parent windows.)</td>
</tr>
<tr>
<td>Filled</td>
<td>Activate to fill the rectangle with the background color.</td>
</tr>
<tr>
<td>Flat</td>
<td>Activate to display a button with a two-dimensional border that indicates the area where the user can click the button. When the cursor is positioned inside this area, a 3D border appears. (Windows only)</td>
</tr>
<tr>
<td>Graphic-Edge</td>
<td>Activate to draw the rectangle edge with a graphic line drawing character (thin line) in character environments. The AppBuilder sets the GRAPHIC-EDGE attribute to TRUE. When you specify values for EDGE-PIXELS greater than three, Graphic Edge is automatically deactivated and the rectangle edge is drawn with a character cell block. See conventions for rules dictating how Graphic-Edge relates to the Edge Pixels field on the RECTANGLE Property Sheet.</td>
</tr>
<tr>
<td>Hidden</td>
<td>Activate to set the HIDDEN attribute to TRUE for windows, frames, dialog boxes, and field-level widgets.</td>
</tr>
<tr>
<td>Horizontal</td>
<td>Activate to set the orientation of the slider or of radio buttons in the radio set to horizontal. The AppBuilder sets the HORIZONTAL attribute to TRUE.</td>
</tr>
<tr>
<td>Keep-Frame-Z-Order</td>
<td>Activate to disable frames automatically moving-to-top when receiving input focus at run time. The AppBuilder sets the KEEP-FRAME-Z-ORDER attribute to TRUE in the CREATE WIDGET statement.</td>
</tr>
<tr>
<td>Keep-Tab-Order</td>
<td>Activate to maintain the existing tab order of field-level objects. The AppBuilder adds the KEEP-TAB-ORDER option to the Frame phrase. You can manually override KEEP-TAB-ORDER for a particular field-level object with the MOVE-AFTER-TAB-ITEM and MOVE-BEFORE-TAB-ITEM methods.</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Large</td>
<td>Activate to set the editor object to edit unlimited text (the limit is your system's memory). The AppBuilder sets the LARGE attribute to TRUE. The small editor provides a limit of 20 KB. (Windows only)</td>
</tr>
<tr>
<td>Large–To–Small</td>
<td>Activate to reverse the default order of values in the slider. When this toggle box is selected, values slide from higher to lower values. By default, values slide from lower to higher.</td>
</tr>
<tr>
<td>Max-Button</td>
<td>Activate to display the maximize button at the right end of the Window's title bar. (Windows only)</td>
</tr>
<tr>
<td>Message-Area</td>
<td>Activate to add a message area to the selected window. The AppBuilder sets the MESSAGE-AREA attribute to TRUE in the CREATE WIDGET statement. (In character environments, all windows have a message area.)</td>
</tr>
<tr>
<td>Min-Button</td>
<td>Activate to display the minimize button at the right end of the Window's title bar. (Windows only)</td>
</tr>
<tr>
<td>Multiple-Selection</td>
<td>Activate to allow the user to choose multiple entries in selection lists and browse objects. The AppBuilder sets the MULTIPLE attribute to TRUE.</td>
</tr>
<tr>
<td>Note:</td>
<td>The user must use row markers to multi-select a browse that has an enabled field. By default, NO-ROW-MARKERS are specified by the AppBuilder. Therefore, you must explicitly turn ROW-MARKERS on to take advantage of the multiple selection feature in an editable browse.</td>
</tr>
<tr>
<td>Native</td>
<td>Activate to make the fill-in behave like a native fill-in field under the current user interface (such as Windows). Otherwise, the field behaves like a default OpenEdge fill-in field under any interface. The AppBuilder adds the NATIVE option to the VIEW-AS phrase.</td>
</tr>
<tr>
<td>No-Assgin</td>
<td>Activate to prevent data assignments to updatable fields from being automatically written back to the database when the user moves to a different row (record). The ABL programmer can write data assignments to the database manually.</td>
</tr>
<tr>
<td>No-Auto-Validate</td>
<td>Activate to turn off, for all field-level objects in the dialog box, the automatic validation defined in the Data Dictionary.</td>
</tr>
<tr>
<td>No-Box</td>
<td>Activate to display no box around the frame. If you do not use this option, a box displays around the frame you are displaying. The AppBuilder adds the NO-BOX option to the Frame phrase.</td>
</tr>
<tr>
<td>No-Current-Value</td>
<td>Activate to display the current value represented on the slider. By default, this toggle box is not selected. (NO CURRENT VALUE = FALSE).</td>
</tr>
<tr>
<td>No-Focus</td>
<td>Activate to prevent a button from taking focus when the mouse is clicked on it. (Windows only)</td>
</tr>
<tr>
<td>No-Help</td>
<td>Activate to disregard all help strings specified in the Data Dictionary for fields entered in the frame. The AppBuilder adds the NO-HELP option to the Frame phrase.</td>
</tr>
<tr>
<td>-------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>No-Hide</td>
<td>Activate to suppress the automatic hiding of the frame (when the block where the frame is scoped iterates). The AppBuilder adds the NO-HIDE option to the Frame phrase.</td>
</tr>
<tr>
<td>No-Labels</td>
<td>Activate to disable any labels from being displayed in the frame or browse. The AppBuilder adds NO-LABELS to the DEFINE FRAME statement for frames and NO-LABELS to the browse phrase of the DEFINE BROWSE statement for browse. (You get column labels when you deactivate NO-LABELS and SIDE-LABELS.)</td>
</tr>
<tr>
<td>No-Row-Markers</td>
<td>Activate to prevent row markers from being displayed in an editable browse. The AppBuilder adds the NO-ROW-MARKERS option to the DEFINE BROWSE statement.</td>
</tr>
<tr>
<td>No-Tab-Stop</td>
<td>Activate to remove the SmartDataBrowser from the applicable tab-traversal list. In Windows, removing the object from the traversal list has some effects that might confuse the user.</td>
</tr>
<tr>
<td>No-Underline</td>
<td>Activate if you do not want to underline labels that appear above fields. The AppBuilder adds the NO-UNDERLINE option to the Frame phrase.</td>
</tr>
<tr>
<td>No-Undo</td>
<td>Activate for the AppBuilder to add the NO-UNDO option to the DEFINE VARIABLE statement for the object. The value of the variable is not restored to its prior value during a backed out transaction or a sub-transaction. NO-UNDO variables are often much more efficient, and you should carefully consider when you can do without the default undo service.</td>
</tr>
<tr>
<td>No-Validate</td>
<td>Activate to disregard all validation conditions specified in the Data Dictionary for the fields entered in this frame. The AppBuilder adds the NO-VALIDATE option to the Frame phrase.</td>
</tr>
<tr>
<td>Open-the-Query</td>
<td>Open a query and populate the frame or browse object in the default enabling procedure.</td>
</tr>
<tr>
<td>Overlay</td>
<td>Activate for the frame to overlay any other frame that does not use the TOP-ONLY option. If you do not use this option, the frame you are using will cause overlapping frames to be hidden. The AppBuilder adds the OVERLAY option to the Frame phrase.</td>
</tr>
<tr>
<td>Read-Only</td>
<td>Activate for the AppBuilder to set the READ-ONLY attribute to TRUE. Editor objects cannot be enabled for input.</td>
</tr>
<tr>
<td>Remove-from-Layout</td>
<td>Activate for the object not to appear in a custom layout.</td>
</tr>
<tr>
<td>Resize</td>
<td>Activate to allow users to resize the window. The AppBuilder sets the RESIZE attribute to TRUE in the CREATE WIDGET statement.</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Retain-Shape</td>
<td>Activate to indicate that the image should retain its aspect ratio (expand or contract equally in both dimensions). Can be set only if Stretch to Fit is set. Does not apply to images that take the <code>.ico</code> file extension.</td>
</tr>
<tr>
<td>Return-Inserted</td>
<td>Activate to cause a RETURN key function event insert a hard return at the cursor position, breaking the current line. The AppBuilder sets the RETURN-INSERTED attribute to TRUE. (Windows only)</td>
</tr>
<tr>
<td>Run-Persistent</td>
<td>Activate to run and create (instantiate) the selected window as a persistent procedure. The AppBuilder runs the window with the PERSISTENT option of the RUN statement.</td>
</tr>
<tr>
<td>Scroll-Bars</td>
<td>Activate to add scroll bars to the selected window. The AppBuilder sets the SCROLL-BARS attribute to TRUE in the CREATE WIDGET statement.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> In Windows, the scroll bars only appear if needed. (When the virtual size of the window exceeds the viewport size.)</td>
</tr>
<tr>
<td>Scrollable</td>
<td>Activate to add scroll bars to the frame. The AppBuilder sets the SCROLLABLE attribute to TRUE.</td>
</tr>
<tr>
<td>Scrollbar-Horizontal</td>
<td>Activate to add a horizontal scrollbar to a selection list or to an editor object. The AppBuilder sets the SCROLLBAR-HORIZONTAL attribute to TRUE and adds the SCROLLBAR-HORIZONTAL option to the Selection List phrase or to the Editor phrase.</td>
</tr>
<tr>
<td>Scrollbar-Vertical</td>
<td>Activate to add a vertical scrollbar to a selection list or to an editor object. The AppBuilder sets the SCROLLBAR-VERTICAL attribute to TRUE and adds the SCROLLBAR-VERTICAL option to the Selection List phrase or to the Editor phrase.</td>
</tr>
<tr>
<td>Sensitive</td>
<td>Activate to allow the user to give input focus to the object. The AppBuilder sets the SENSITIVE attribute for the object to TRUE.</td>
</tr>
<tr>
<td>Separators</td>
<td>Activate to add rules between rows and columns in a browse. The AppBuilder adds the SEPARATORS option to the DEFINE BROWSE statement.</td>
</tr>
<tr>
<td>Shared</td>
<td>Activate to share the object among several procedures. Depending upon the object type, the AppBuilder adds <code>{&amp;NEW}SHARED</code> to the object definition.</td>
</tr>
<tr>
<td>Show-in-Taskbar</td>
<td>Activate to make the Window icon appear in the taskbar and in the task-switching window. (Windows only)</td>
</tr>
<tr>
<td>Side-Labels</td>
<td>Activate to specify side labels for the fields in the frame. The AppBuilder adds the SIDE-LABELS option to the Frame phrase. When deactivated, column labels are displayed above their corresponding fields.</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Size-to-Fit</td>
<td>Activate to specify that the frame be sized just large enough to fit the basic objects and SmartObject instances it contains. The AppBuilder removes the explicit definition of the frame size from the Frame phrase. A change to this attribute will only be noted upon saving and reloading the design window, or at runtime.</td>
</tr>
<tr>
<td>Small-Title</td>
<td>Activate to reduce the size of the Window’s title bar to the minimum needed for the title, as well as disable the maximize and minimize buttons at the right end of the title bar. Windows with small title bars do not have maximize or minimize buttons; they only have close buttons.</td>
</tr>
<tr>
<td>Sort</td>
<td>Activate to insert all items in the selection list in sorted order (overriding any other sort methods: ADD-FIRST, ADD-LAST, INSERT, and REPLACE). The AppBuilder sets the SORT attribute to TRUE.</td>
</tr>
<tr>
<td>Status-Area</td>
<td>Activate to add a status area to the selected window. The AppBuilder sets the STATUS-AREA attribute to TRUE in the CREATE WIDGET statement. (In character environments, all windows have a status area.)</td>
</tr>
<tr>
<td>Stretch-to-Fit</td>
<td>Activate to stretch the image to fit the available space. If Retain Shape is also set, the stretched image will maintain its original aspect ratio, even at the expense of fitting the space. Does not apply to images that take the .ico file extension.</td>
</tr>
<tr>
<td>Suppress Window</td>
<td>Activate to convert the window to “frame-only.” When run, frames appear in the CURRENT-WINDOW. The AppBuilder comments out the CREATE WINDOW statement.</td>
</tr>
<tr>
<td>Title-Bar</td>
<td>Activate to specify a title for frames, dialog boxes, browse objects, and pop-up menus. The AppBuilder adds the NO-BOX option to the Frame phrase.</td>
</tr>
<tr>
<td>Transparent</td>
<td>Activate to make the background color of an image transparent, thus revealing whatever is underneath it. The background color is determined by the color of the pixel in the lower-left corner of the image. Does not apply to images that take the .ico file extension.</td>
</tr>
<tr>
<td>Use-Dict-Exps</td>
<td>Activate to ensure that validation expressions and help strings from the Data Dictionary are compiled into the application for all objects in the frame. The AppBuilder adds the USE-DICT-EXPS option to the Frame phrase.</td>
</tr>
<tr>
<td>View</td>
<td>Activate to cause the frame to be viewed at startup. (This toggle box is useful only when you have set the frame to HIDDEN.)</td>
</tr>
<tr>
<td>View-As-Text</td>
<td>Activate to make the fill-in field a read-only TEXT object. In a graphical environment, a text field takes up less space on the screen because it does not have the decorations associated with a native fill-in field. The AppBuilder adds the TEXT option to the VIEW-AS phrase.</td>
</tr>
<tr>
<td>Visible</td>
<td>Activate to set the VISIBLE attribute to TRUE.</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>Word-Wrap</td>
<td>Activate to make the editor object automatically break lines at any word that crosses the word wrap margin of the text area. The AppBuilder sets the WORD-WRAP attribute to TRUE.</td>
</tr>
</tbody>
</table>

**Note:** See the ABL Reference help for information about the statements, attributes, and phrases listed above.

## Advanced Properties Sheet dialogs

The Advanced Properties dialog allows you to modify the advanced properties of a selected object. Access this dialog by clicking **Advanced** on the Property Sheet dialog.

### Advanced properties for Window

The Advanced Properties dialog allows you to set advanced properties for the selected window.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Private Data</th>
<th>Enter private data about the selected window. Private data is an annotation containing any character data that you can attach to an object. This annotation is only informational and not used by the code. The GUI Designer reads and writes private data as comments in the Run-time Attribute and AppBuilder Settings section of procedure files.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Characters</td>
<td>Select this option to write window size and position in character units. The GUI Designer writes the values entered in Column, Row, Width, and Height on the WINDOW Property Sheet in character units in the CREATE WIDGET statement.</td>
</tr>
<tr>
<td>Pixels</td>
<td>Select this option to express window size and position in pixel units. The GUI Designer writes the values entered in X, Y, Width Pixels, and Height Pixels in pixel units in the CREATE WIDGET statement.</td>
</tr>
<tr>
<td>Custom Lists</td>
<td>Select the custom list check box to add the window name to the selected custom list. For example, select the {&amp;List-1} check box to add the window name to custom list &amp;List-2. One of six user-defined, alphabetical lists of space-delimited object names represented by the preprocessor names (&amp;List-1, &amp;List-2, &amp;List-3, &amp;List-4, &amp;List-5, &amp;List-6). You can use custom lists in ABL code whenever you need to represent object names, as in the following example: ENABLE ALL EXCEPT {&amp;List-1} WITH FRAME {&amp;FRAME-NAME}. HIDE {&amp;LIST-2} WITH FRAME {&amp;FRAME-NAME}. The GUI Designer adds an object name to a custom list whenever you activate one of the list toggles on the object's Advanced Property Sheet. Objects that are cut or deleted from a procedure file are automatically removed from the appropriate list.</td>
</tr>
<tr>
<td>X</td>
<td>Enter a value in pixels from your monitor left edge where you want to place the window upper left corner.</td>
</tr>
<tr>
<td>Y</td>
<td>Enter a value in pixels from your monitor top edge where you want to place the window upper left corner.</td>
</tr>
<tr>
<td>Width Pixels</td>
<td>Enter a value for the number of pixel units wide to make the window.</td>
</tr>
</tbody>
</table>
### Advanced properties for Dialog Box

The **Advanced Properties** dialog allows you to set advanced properties for the selected dialog box.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Height Pixels</strong></td>
<td>Enter a value for the number of pixel units high to make the window.</td>
</tr>
<tr>
<td><strong>Virtual Width Pixels</strong></td>
<td>Enter the maximum and virtual width (in pixel units) of the window.</td>
</tr>
<tr>
<td><strong>Virtual Height Pixels</strong></td>
<td>Enter the maximum and virtual height (in pixel units) of the window.</td>
</tr>
<tr>
<td><strong>Current Layout</strong></td>
<td>Displays the current layout used to display the window.</td>
</tr>
<tr>
<td><strong>Sync With Master</strong></td>
<td>Displays the <strong>Sync With Master</strong> dialog which allows you to overwrite the current layout with the window default properties.</td>
</tr>
</tbody>
</table>

**Note:** The **Sync With Master** option is enabled if you have defined a master layout.

For more information, see Property Sheet dialogs on page 702

### Advanced properties for Dialog Box

The **Advanced Properties** dialog allows you to set advanced properties for the selected dialog box.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Private Data</strong></td>
<td>Enter a private data about the selected dialog box. Private data is an annotation containing any character data that you can attach to an object. This annotation is only informational and not used by the code. The GUI Designer reads and writes private data as comments in the Run-time Attribute and AppBuilder Settings section of procedure files.</td>
</tr>
<tr>
<td><strong>Characters</strong></td>
<td>Select this option to write dialog box size and position in character units. The GUI Designer writes the values entered in Column, Row, Width, and Height on the DIALOG-BOX Property Sheet in character units using the Size phrase in the DEFINE FRAME statement.</td>
</tr>
<tr>
<td><strong>Pixels</strong></td>
<td>Select this option to express dialog box size and position in pixel units. The GUI Designer writes the values entered in X, Y, Width Pixels, and Height Pixels in pixel units using the Size and At phrases in the DEFINE VARIABLE and DEFINE FRAME statements.</td>
</tr>
</tbody>
</table>
| **Custom Lists**  | Select the custom list check box to add the dialog box name to the selected custom list. For example, select the \&List-1 check box to add the dialog box name to custom list \&List-2. One of six user-defined, alphabetical lists of space-delimited object names represented by the preprocessor names (\&List-1, \&List-2, \&List-3,\&List-4, \&List-5, \&List-6). You can use custom lists in ABL code whenever you need to represent object names, as in the following example:  

```ABL
ENABLE ALL EXCEPT \&LIST-1 WITH FRAME \&FRAME-NAME.
HIDE \&LIST-2 WITH FRAME \&FRAME-NAME.
```

The GUI Designer adds an object name to a custom list whenever you activate one of the list toggles on the object's Advanced Property Sheet. Objects that are cut or deleted from a procedure file are automatically removed from the appropriate list. |
Enter a window column value (in pixel) where you want to place the dialog box upper left corner relative to the default AppBuilder run-time window. This option is enabled only when the Explicit Position toggle box on the DIALOG-BOX Property Sheet is selected.

**Note:** This attribute only takes effect at run time.

Enter a window row value (in pixels) where you want to place the dialog box upper left corner relative to the default AppBuilder run-time window. This option is enabled only when the Explicit Position toggle box on the DIALOG-BOX Property Sheet is selected.

**Note:** This attribute only takes effect at run time.

Enter a value for the number of pixel units wide to make the dialog box.

Enter a value for the number of pixel units high to make the dialog box.

Select this option to choose one or more objects in the frame or dialog box by stretching a select box around the objects at run-time. The GUI Designer sets the for the frame or the dialog box to TRUE.

Displays the current layout used to display the dialog box.

Displays the Sync with Master dialog on page 782 which allows you to overwrite the current layout with the dialog box default properties.

**Note:** The Sync With Master option is enabled if you have defined a master layout.

For more information, see Property Sheet dialogs on page 702

**Advanced properties for Browse**

The Advanced Properties - Browse dialog allows you to set advanced properties for the selected browse widget.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Help</th>
<th>Enter a help string for the selected browse widget. The GUI Designer displays the help string in the run-time window status bar whenever the browse has input focus.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private Data</td>
<td>Enter any private data about the selected browse widget. Private data is an annotation containing any character data that you can attach to an object. This annotation is only informational and not used by the code. The GUI Designer reads and writes private data as comments in the Run-time Attribute and AppBuilder Settings section of procedure files.</td>
</tr>
<tr>
<td>Characters</td>
<td>Select this option to write browse size and position in character units. The GUI Designer writes the values entered in Column, Row, Width, and Height on the BROWSE Property Sheet in character units using the Size and At phrases in the DEFINE BROWSE and DEFINE FRAME statements.</td>
</tr>
<tr>
<td>Pixels</td>
<td>Select this option to write browse size and position in pixel units. The GUI Designer writes the values entered in X, Y, Width Pixels, and Height Pixels in pixel units using the Size and At phrases in the DEFINE BROWSE and DEFINE FRAME statements.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>
| Custom Lists | Select the custom list check box to add the browse name to the selected custom list. For example, select the {&List-1} check box to add the browse name to custom list &List-2. One of six user-defined, alphabetical lists of space-delimited object names represented by the preprocessor names (&List-1, &List-2, &List-3, &List-4, &List-5, &List-6). You can use custom lists in ABL code whenever you need to represent object names, as in the following example: 
ENABLE ALL EXCEPT {&LIST-1} WITH FRAME {&FRAME-NAME}. HIDE {&LIST-2} WITH FRAME {&FRAME-NAME}. The GUI Designer adds an object name to a custom list whenever you activate one of the list toggles on the object's Advanced Property Sheet. Objects that are cut or deleted from a procedure file are automatically removed from the appropriate list. |
| X | Enter a value in pixels from the window left edge where you want to place the browse upper left or right corner.  
**Note:** Depends upon the position setting. For example, if Left-Align is chosen, the value in X refers to the browse upper left corner. |
| Y | Enter a value in pixels from the window top edge where you want to place the browse upper left or right corner.  
**Note:** Depends upon the position setting. For example, if Left-Align is chosen, the value in Y refers to the browse upper left corner. |
| Width Pixels | Enter a value for the number of pixel units wide to make the browse. |
| Height Pixels | Enter a value for the number of pixel units high to make the browse height. |
| Row Height Pixels | Enter a value for the number of pixel units high to make the row height in the selected browse. |
| Left-Align | Select this option to position the browse left edge along the column / X position specified. |
| Right-Align | Select this option to position the browse right edge along the column / X position specified. |
| Advanced Settings | This section allows you to select advanced run-time properties and other ABL related options for the selected browse widget. |
### Current Layout
Displays the current layout used to display the browse.

### Sync with Master
Displays the **Sync With Master** dialog which allows you to overwrite the current layout with the browse default properties.

**Note:** The **Sync With Master** option is enabled if you have defined a master layout.

---

For more information, see [Property Sheet dialogs](#) on page 702

### Advanced properties for Frame

The **Advanced Properties** dialog allows you to modify the advanced properties for the selected frame.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Private Data</strong></td>
<td>Enter any private data about the selected frame widget. Private data is an annotation containing any character data that you can attach to an object. This annotation is only informational and not used by the code. The GUI Designer reads and writes private data as comments in the Run-time Attribute and AppBuilder Settings section of procedure files.</td>
</tr>
<tr>
<td><strong>Characters</strong></td>
<td>Select this option to write frame size and position in character units. The GUI Designer writes the values entered in Column, Row, Width, and Height on the EDITOR Property Sheet in character units using the Size and At phrases in the DEFINE FRAME statement.</td>
</tr>
<tr>
<td><strong>Pixels</strong></td>
<td>Select this option to express frame size and position in pixel units. The GUI Designer writes the values entered in X, Y, Width Pixels, and Height Pixels in pixel units using the Size and At phrases in the DEFINE FRAME statement.</td>
</tr>
</tbody>
</table>
| **Custom Lists**| Select the custom list check box to add the frame name to the selected custom list. For example, select the {&List-1} check box to add the frame name to custom list &List-2. One of six user-defined, alphabetical lists of space-delimited object names represented by the preprocessor names (&List-1, &List-2, &List-3, &List-4, &List-5, &List-6). You can use custom lists in ABL code whenever you need to represent object names, as in the following example:  

```
ENABLE ALL EXCEPT {&LIST-1} WITH FRAME {&FRAME-NAME}.

HIDE {&LIST-2} WITH FRAME {&FRAME-NAME}.
```

The GUI Designer adds an object name to a custom list whenever you activate one of the list toggles on the object's Advanced Property Sheet. Objects that are cut or deleted from a procedure file are automatically removed from the appropriate list. |
<table>
<thead>
<tr>
<th><strong>X</strong></th>
<th>Enter a value in pixels from the window left edge where you want to place the frame upper left or right corner.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>Depends upon the position setting. For example, if Left-Align is chosen, the value in X refers to the browse upper left corner.</td>
</tr>
<tr>
<td><strong>Y</strong></td>
<td>Enter a value in pixels from the window top edge where you want to place the frame upper left or right corner.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>Depends upon the position setting. For example, if Left-Align is chosen, the value in Y refers to the browse upper left corner.</td>
</tr>
<tr>
<td><strong>Width Pixels</strong></td>
<td>Enter a value for the number of pixel units wide to make the frame.</td>
</tr>
<tr>
<td><strong>Height Pixels</strong></td>
<td>Enter a value for the number of pixel units high to make the frame height.</td>
</tr>
<tr>
<td><strong>Virtual Width Pixels</strong></td>
<td>Enter a value for the number of pixel units wide to make the row width in the selected frame.</td>
</tr>
<tr>
<td><strong>Virtual Height Pixels</strong></td>
<td>Enter a value for the number of pixel units high to make the row height in the selected frame.</td>
</tr>
<tr>
<td><strong>Advanced Settings</strong></td>
<td>This section allows you to select advanced run-time properties and other ABL related options for the selected frame.</td>
</tr>
<tr>
<td><strong>Current Layout</strong></td>
<td>Displays the current layout used to display the frame.</td>
</tr>
<tr>
<td><strong>Sync with Master</strong></td>
<td>Displays the <strong>Sync With Master</strong> dialog which allows you to overwrite the current layout with the frame default properties.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>The <strong>Sync With Master</strong> option is enabled if you have defined a master layout.</td>
</tr>
</tbody>
</table>

For more information, see *Property Sheet dialogs* on page 702

**Advanced properties for Rectangle**

The **Advanced Properties** dialog allows you to set advanced properties for the selected rectangle.

This dialog includes the following options:

| **Private Data** | Enter any private data about the selected browse widget. Private data is an annotation containing any character data that you can attach to an object. This annotation is only informational and not used by the code. The GUI Designer reads and writes private data as comments in the Run-time Attribute and AppBuilder Settings section of procedure files. |
| **Characters** | Select this option to write rectangle size and position in character units. The AppBuilder writes the values entered in Column, Row, Width, and Height on the BUTTON Property Sheet in character units using the Size and At phrases in the DEFINE RECTANGLE and DEFINE FRAME statements. |
| **Pixels** | Select this option to write rectangle size and position in pixel units. The AppBuilder writes the values entered in X, Y, Width Pixels, and Height Pixels in pixel units using the Size and At phrases in the DEFINE RECTANGLE and DEFINE FRAME statements. |
| **Custom Lists** | Select the custom list check box to add the rectangle name to the selected custom list. For example, select the {&List-1} check box to add the rectangle name to custom list &List-2. One of six user-defined, alphabetical lists of space-delimited object names represented by the preprocessor names (&List-1, &List-2, &List-3,&List-4, &List-5, &List-6). You can use custom lists in ABL code whenever you need to represent object names, as in the following example:  
DISABLE ALL EXCEPT {&LIST-1} WITH FRAME {&FRAME-NAME}.  
DISABLE {&LIST-2} WITH FRAME {&FRAME-NAME}.  
The GUI Designer adds an object name to a custom list whenever you activate one of the list toggles on the object's Advanced Property Sheet. Objects that are cut or deleted from a procedure file are automatically removed from the appropriate list. |
| **X** | Enter a value in pixels from the window left edge where you want to place the rectangle upper left or right corner.  
**Note:** Depends upon the position setting. For example, if Left-Align is chosen, the value in X refers to the rectangle upper left corner. |
| **Y** | Enter a value in pixels from the window left edge where you want to place the rectangle upper left or right corner.  
**Note:** Depends upon the position setting. For example, if Left-Align is chosen, the value in X refers to the rectangle upper left corner. |
| **Width Pixels** | Enter a value for the number of pixel units wide to make the rectangle. |
| **Height Pixels** | Enter a value for the number of pixel units high to make the rectangle. |
| **Left-Align** | Select this option to position the rectangle left edge along the column / X position specified. |
| **Right-Align** | Select this option to position the rectangle right edge along the column / X position specified. |
| **Advanced Settings** | This section allows you to select advanced run-time properties and other ABL related options for the selected rectangle. |
**Current Layout**
Displays the current layout used to display the rectangle.

**Sync with Master**
Displays the Sync With Master dialog which allows you to overwrite the current layout with the rectangle default properties.

*N*ote: The Sync With Master option is enabled if you have defined a master layout.

For more information, see Property Sheet dialogs on page 702

**Advanced properties for Image**

The Advanced Properties dialog allows you to set advanced properties for the selected image.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Private Data</strong></td>
<td>Enter any private data about the selected image. Private data is an annotation containing any character data that you can attach to an object. This annotation is only informational and not used by the code. The GUI Designer reads and writes private data as comments in the Run-time Attribute and AppBuilder Settings section of procedure files.</td>
</tr>
<tr>
<td><strong>Characters</strong></td>
<td>Select this option to write image size and position in character units. The AppBuilder writes the values entered in Column, Row, Width, and Height on the EDITOR Property Sheet in character units using the Size and At phrases in the DEFINE IMAGE and DEFINE FRAME statements.</td>
</tr>
<tr>
<td><strong>Pixels</strong></td>
<td>Select this option to write image size and position in pixel units. The AppBuilder writes the values entered in X, Y, Width Pixels, and Height Pixels in pixel units using the Size and At phrases in the DEFINE FRAME statement.</td>
</tr>
<tr>
<td><strong>Custom Lists</strong></td>
<td>Select the custom list check box to add the image name to the selected custom list. For example, select the {&amp;List-1} check box to add the image name to custom list &amp;List-2. One of six user-defined, alphabetical lists of space-delimited object names represented by the preprocessor names (&amp;List-1, &amp;List-2, &amp;List-3,&amp;List-4, &amp;List-5, &amp;List-6). You can use custom lists in ABL code whenever you need to represent object names, as in the following example: ENABLE ALL EXCEPT {&amp;LIST-1} WITH FRAME {&amp;FRAME-NAME}. HIDE {&amp;LIST-2} WITH FRAME {&amp;FRAME-NAME}. The GUI Designer adds an object name to a custom list whenever you activate one of the list toggles on the object's Advanced Property Sheet. Objects that are cut or deleted from a procedure file are automatically removed from the appropriate list.</td>
</tr>
<tr>
<td><strong>X</strong></td>
<td>Enter a value in pixels from the window left edge where you want to place the image upper left corner.</td>
</tr>
<tr>
<td><strong>Y</strong></td>
<td>Enter a value in pixels from the window top edge where you want to place the image upper left corner.</td>
</tr>
<tr>
<td><strong>Width Pixels</strong></td>
<td>Enter a value for the number of pixel units wide to make the image.</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Height Pixels</strong></td>
<td>Enter a value for the number of pixel units high to make the image.</td>
</tr>
<tr>
<td><strong>Left-Align</strong></td>
<td>Select this option to position the image left edge along the column / X position specified.</td>
</tr>
<tr>
<td><strong>Right-Align</strong></td>
<td>Select this option to position the image right edge along the column / X position specified.</td>
</tr>
<tr>
<td><strong>Advanced Settings</strong></td>
<td>This section allows you to select advanced run-time properties and other ABL related options for the selected image.</td>
</tr>
<tr>
<td><strong>Current Layout</strong></td>
<td>Displays the current layout used to display the image.</td>
</tr>
<tr>
<td><strong>Sync with Master</strong></td>
<td>Displays the <a href="#">Sync With Master</a> dialog that allows you to overwrite the current layout with the image default properties.</td>
</tr>
</tbody>
</table>

**Note:** The Sync With Master option is enabled if you have defined a master layout.

For more information, see [Property Sheet dialogs](#) on page 702

### Advanced properties for Radio Set

The Advanced Properties dialog allows you to set advanced properties for the selected radio set.

This dialog includes the following options:

<table>
<thead>
<tr>
<th><strong>Initial Value</strong></th>
<th>Enter an initial value for the radio set. The GUI Designer writes this value to the DEFINE VARIABLE statement for the radio set.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Help</strong></td>
<td>Enter a help string for the radio set. The GUI Designer displays the help string in the run-time window status bar whenever the radio set has input focus.</td>
</tr>
<tr>
<td><strong>Private Data</strong></td>
<td>Enter any private data about the radio set. Private data is an annotation containing any character data that you can attach to an object. This annotation is only informational and not used by the code. The GUI Designer reads and writes private data as comments in the Run-time Attribute and AppBuilder Settings section of procedure files.</td>
</tr>
<tr>
<td><strong>Characters</strong></td>
<td>Select this option to write radio set size and position in character units. The AppBuilder writes the values entered in Column, Row, Width, and Height on the BUTTON Property Sheet in character units using the Size and At phrases in the DEFINE VARIABLE and DEFINE FRAME statements.</td>
</tr>
<tr>
<td><strong>Pixels</strong></td>
<td>Select this option to write radio set size and position in pixel units. The AppBuilder writes the values entered in X, Y, Width Pixels, and Height Pixels in pixel units using the Size and At phrases in the DEFINE VARIABLE and DEFINE FRAME statements.</td>
</tr>
</tbody>
</table>
Custom Lists

Select this option to add the radio set name to the selected custom list. For example, select the `{&List-1}` check box to add the radio set name to custom list `{&List-2}`.

One of six user-defined, alphabetical lists of space-delimited object names represented by the preprocessor names (`&List-1`, `&List-2`, `&List-3`, `&List-4`, `&List-5`, `&List-6`). You can use custom lists in ABL code whenever you need to represent object names, as in the following example:

```
ENABLE ALL EXCEPT `{&LIST-1}` WITH FRAME `{&FRAME-NAME}`.
HIDE `{&LIST-2}` WITH FRAME `{&FRAME-NAME}`.
```

The GUI Designer adds an object name to a custom list whenever you activate one of the list toggles on the object's Advanced Property Sheet. Objects that are cut or deleted from a procedure file are automatically removed from the appropriate list.

<table>
<thead>
<tr>
<th>X</th>
<th>Enter a value in pixels from the window left edge where you want to place the radio set upper left or right corner.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Note:</strong> Depends upon the position setting. For example, if Left-Align is chosen, the value in X refers to the radio set upper left corner.</td>
</tr>
<tr>
<td>Y</td>
<td>Enter a value in pixels from the window top edge where you want to place the radio set upper left or right corner.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Depends upon the position setting. For example, if Left-Align is chosen, the value in Y refers to the radio set upper left corner.</td>
</tr>
<tr>
<td>Width Pixels</td>
<td>Enter a value for the number of pixel units wide to make the radio set.</td>
</tr>
<tr>
<td>Height Pixels</td>
<td>Enter a value for the number of pixel units high to make the radio set.</td>
</tr>
<tr>
<td>Left-Align</td>
<td>Select this option to position the radio set left edge along the column / X position specified.</td>
</tr>
<tr>
<td>Right-Align</td>
<td>Select this option to position the radio set right edge along the column / X position specified.</td>
</tr>
<tr>
<td>Advanced Settings</td>
<td>This section allows you to select advanced run-time properties and other ABL related options for the selected radio set.</td>
</tr>
<tr>
<td>Current Layout</td>
<td>Displays the current layout used to display the radio set.</td>
</tr>
<tr>
<td>Sync With Master</td>
<td>Displays the Sync With Master dialog which allows you to overwrite the current layout with the radio set default properties.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The Sync With Master option is enabled if you have defined a master layout.</td>
</tr>
</tbody>
</table>

For more information, see Property Sheet dialogs on page 702

**Advanced properties for Toggle Box**

The Advanced Properties dialog allows you to set advanced properties for the selected toggle box.
This dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Value</td>
<td>Enter an initial value for the toggle box. The GUI Designer writes this value to the DEFINE VARIABLE statement for the toggle box.</td>
</tr>
<tr>
<td>Help</td>
<td>Enter a help string for the toggle box. The GUI Designer displays the help string in the run-time window status bar whenever the toggle box has input focus.</td>
</tr>
<tr>
<td>Private Data</td>
<td>Enter any private data about the toggle box. Private data is an annotation containing any character data that you can attach to an object. This annotation is only informational and not used by the code. The GUI Designer reads and writes private data as comments in the Run-time Attribute and AppBuilder Settings section of procedure files.</td>
</tr>
<tr>
<td>Characters</td>
<td>Select this option to write toggle box size and position in character units. The GUI Designer writes the values entered in Column, Row, Width, and Height on the TOGGLE-BOX Property Sheet in character units using the Size and At phrases in the DEFINE VARIABLE and DEFINE FRAME statements.</td>
</tr>
<tr>
<td>Pixels</td>
<td>Select this option to write toggle box size and position in pixel units. The GUI Designer writes the values entered in X, Y, Width Pixels, and Height Pixels in pixel units using the Size and At phrases in the DEFINE VARIABLE and DEFINE FRAME statements.</td>
</tr>
</tbody>
</table>
| Custom Lists    | Select this option to add the toggle box name to the selected custom list. For example, select the {&List-1} check box to add the toggle box name to custom list &List-2. One of six user-defined, alphabetical lists of space-delimited object names represented by the preprocessor names (&List-1, &List-2, &List-3, &List-4, &List-5, &List-6). You can use custom lists in ABL code whenever you need to represent object names, as in the following example:  

```
ENABLE ALL EXCEPT {&LIST-1} WITH FRAME {&FRAME-NAME}.
```

```
HIDE {&LIST-2} WITH FRAME {&FRAME-NAME}.
```

The GUI Designer adds an object name to a custom list whenever you activate one of the list toggles on the object's Advanced Property Sheet. Objects that are cut or deleted from a procedure file are automatically removed from the appropriate list. |
| X               | Enter a value in pixels from the window left edge where you want to place the toggle box upper left or right corner.                     |
| Note:           | Depends upon the position setting. For example, if Left-Align is chosen, the value in X refers to the toggle box upper left corner.     |
| Y               | Enter a value in pixels from the window top edge where you want to place the toggle box upper left or right corner.                   |
| Note:           | Depends upon the position setting. For example, if Left-Align is chosen, the value in Y refers to the toggle box upper left corner.     |
| Width Pixels    | Enter a value for the number of pixel units wide to make the toggle box.                                                              |
| Height Pixels   | Enter a value for the number of pixel units high to make the toggle box.                                                              |
Left-Align | Select this option to position the toggle box left edge along the column / X position specified.
---|---
Right-Align | Select this option to position the toggle box right edge along the column / X position specified.
Advanced Settings | This section allows you to select advanced run-time attributes and other ABL related options for the selected toggle box.
Current Layout | Displays the current layout used to display the toggle box.
Sync With Master | Displays the **Sync With Master** dialog which allows you to overwrite the current layout with the toggle box default properties.

**Note:** The **Sync With Master** option is enabled if you have defined a master layout.

For more information, see [Property Sheet dialogs](#) on page 702

**Advanced properties for Slider**

The **Advanced Properties** dialog allows you to set additional properties for the selected slider.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Initial Value</th>
<th>Enter an initial value for the slider. The GUI Designer writes this value to the <code>DEFINE VARIABLE</code> statement for the slider.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Help</td>
<td>Enter a help string for the slider. The GUI Designer displays the help string in the run-time window status bar whenever the slider has input focus.</td>
</tr>
<tr>
<td>Private Data</td>
<td>Enter a private data about the slider. Private data is an annotation containing any character data that you can attach to an object. This annotation is only informational and not used by the code. The GUI Designer reads and writes private data as comments in the Run-time Attribute and AppBuilder Settings section of procedure files.</td>
</tr>
<tr>
<td>Characters</td>
<td>Select this option to write slider size and position in character units. The GUI Designer writes the values entered in Column, Row, Width, and Height on the SLIDER Property Sheet in character units using the Size and At phrases in the <code>DEFINE VARIABLE</code> and <code>DEFINE FRAME</code> statements.</td>
</tr>
<tr>
<td>Pixels</td>
<td>Select this option to write slider size and position in pixel units. The GUI Designer writes the values entered in X, Y, Width Pixels, and Height Pixels in pixel units using the Size and At phrases in the <code>DEFINE VARIABLE</code> and <code>DEFINE FRAME</code> statements.</td>
</tr>
</tbody>
</table>
Select this option to add the slider name to the selected custom list. For example, select the {&List-1} check box to add the slider name to custom list &List-2.

One of six user-defined, alphabetical lists of space-delimited object names represented by the preprocessor names (&List-1, &List-2, &List-3, &List-4, &List-5, &List-6). You can use custom lists in ABL code whenever you need to represent object names, as in the following example:

```abl
ENABLE ALL EXCEPT {&LIST-1} WITH FRAME {&FRAME-NAME}.
HIDE {&LIST-2} WITH FRAME {&FRAME-NAME}.
```

The GUI Designer adds an object name to a custom list whenever you activate one of the list toggles on the object's Advanced Property Sheet. Objects that are cut or deleted from a procedure file are automatically removed from the appropriate list.

<table>
<thead>
<tr>
<th>Custom Lists</th>
<th>Select this option to add the slider name to the selected custom list. For example, select the {&amp;List-1} check box to add the slider name to custom list &amp;List-2.</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Enter a value in pixels from the window left edge where you want to place the slider upper left or right corner. <strong>Note:</strong> Depends upon the position setting. For example, if Left-Align is chosen, the value in X refers to the slider upper left corner.</td>
</tr>
<tr>
<td>Y</td>
<td>Enter a value in pixels from the window top edge where you want to place the slider upper left or right corner. <strong>Note:</strong> Depends upon the position setting. For example, if Left-Align is chosen, the value in Y refers to the slider upper left corner.</td>
</tr>
<tr>
<td>Width Pixels</td>
<td>Enter a value for the number of pixel units wide to make the slider.</td>
</tr>
<tr>
<td>Height Pixels</td>
<td>Enter a value for the number of pixel units high to make the slider.</td>
</tr>
<tr>
<td>Left-Align</td>
<td>Select this option to position the slider left edge along the column / X position specified.</td>
</tr>
<tr>
<td>Right-Align</td>
<td>Select this option to position the slider right edge along the column / X position specified.</td>
</tr>
<tr>
<td>Advanced Settings</td>
<td>This section allows you to select advanced run-time attributes and other ABL related options.</td>
</tr>
<tr>
<td>Current Layout</td>
<td>Displays the current layout used to display the slider.</td>
</tr>
<tr>
<td>Sync With Master</td>
<td>Displays the Sync with Master dialog on page 782 dialog which allows you to overwrite the current layout with the slider default properties. <strong>Note:</strong> The Sync With Master option is enabled if you have defined a master layout.</td>
</tr>
</tbody>
</table>

For more information, see Property Sheet dialogs on page 702
## Advanced properties for Button

The **Advanced Properties** dialog allows you to set advanced properties for the selected button.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Help</td>
<td>Enter a help string for the button. The GUI Designer displays the help string in the run-time window status bar whenever the button has input focus.</td>
</tr>
<tr>
<td>Private Date</td>
<td>Enter any private data about the button. Private data is an annotation containing any character data that you can attach to an object. This annotation is only informational and not used by the code. The GUI Designer reads and writes private data as comments in the Run-time Attribute and AppBuilder Settings section of procedure files.</td>
</tr>
<tr>
<td>Character</td>
<td>Select this option to write button size and position in character units. The GUI Designer writes the values entered in Column, Row, Width, and Height on the BUTTON Property Sheet in character units using the Size and At phrases in the DEFINE BUTTON and DEFINE FRAME statements.</td>
</tr>
<tr>
<td>Pixels</td>
<td>Select this option to write button size and position in pixel units. The GUI Designer writes the values entered in X, Y, Width Pixels, and Height Pixels in pixel units using the Size and At phrases in the DEFINE BUTTON and DEFINE FRAME statements.</td>
</tr>
</tbody>
</table>
| Custom Lists | Select this option to add the button name to the selected custom list. For example, select the \{&List-1\} check box to add the button name to custom list &List-2. One of six user-defined, alphabetical lists of space-delimited object names represented by the preprocessor names (&List-1, &List-2, &List-3, &List-4, &List-5, &List-6). You can use custom lists in ABL code whenever you need to represent object names, as in the following example: ENABLE ALL EXCEPT \{&LIST-1\} WITH FRAME \{&FRAME-NAME\}. HIDE \{&LIST-2\} WITH FRAME \{&FRAME-NAME\}.  

The GUI Designer adds an object name to a custom list whenever you activate one of the list toggles on the object's Advanced Property Sheet. Objects that are cut or deleted from a procedure file are automatically removed from the appropriate list. |
| X            | Enter a value in pixels from the window left edge where you want to place the button upper left or right corner.                                                                                             |
| Y            | Enter a value in pixels from the window top edge where you want to place the button upper left or right corner.                                                                                             |
| Width Pixels | Enter a value for the number of pixel units wide to make the button.                                                                                                                                       |
Height Pixels | Enter a value for the number of pixel units high to make the button.
Left-Align | Select this option to position the button left edge along the column / X position specified.
Right-Align | Select this option to position the button right edge along the column / X position specified.
Advanced Settings | This section allows you to select advanced run-time attributes and other ABL related options.
Current Layout | Displays the current layout used to display the button.
Sync With Master | Displays the Sync With Master dialog which allows you to overwrite the current layout with the button default properties.

**Note:** The Sync With Master option is enabled if you have defined a master layout.

For more information, see Property Sheet dialogs on page 702

**Advanced properties for Selection List**

The Advanced Properties dialog allows you to set advanced properties for the selected selection list.

This dialog includes the following options:

| Initial Value | Enter an initial value for the selection list. The GUI Designer writes this value to the DEFINE VARIABLE statement for the selection list. |
| Help | Enter a help string for the selection list. The GUI Designer displays the help string in the run-time window status bar whenever the selection list has input focus. |
| Private Data | Enter any private data about the selection list. Private data is an annotation containing any character data that you can attach to an object. This annotation is only informational and not used by the code. The GUI Designer reads and writes private data as comments in the Run-time Attribute and AppBuilder Settings section of procedure files. |
| Characters | Select this option to write selection list size and position in character units. The GUI Designer writes the values entered in Column, Row, Width, and Height on the SELECTION-LIST Property Sheet in character units using the Size and At phrases in the DEFINE VARIABLE and DEFINE FRAME statements. |
| Pixels | Select this option to write selection list size and position in pixel units. The GUI Designer writes the values entered in X, Y, Width Pixels, and Height Pixels in pixel units using the Size and At phrases in the DEFINE VARIABLE and DEFINE FRAME statements. |
### Custom Lists

Select this option to add the selection list name to the selected custom list. For example, select the `{&List-1}` check box to add the selection list name to custom list `&List-2`.

One of six user-defined, alphabetical lists of space-delimited object names represented by the preprocessor names (`&List-1`, `&List-2`, `&List-3`, `&List-4`, `&List-5`, `&List-6`). You can use custom lists in ABL code whenever you need to represent object names, as in the following example:

```abl
ENABLE ALL EXCEPT {&LIST-1} WITH FRAME {&FRAME-NAME}.
HIDE {&LIST-2} WITH FRAME {&FRAME-NAME}.
```

The GUI Designer adds an object name to a custom list whenever you activate one of the list toggles on the object's Advanced Property Sheet. Objects that are cut or deleted from a procedure file are automatically removed from the appropriate list.

### X

Enter a value in pixels from the window left edge where you want to place the selection list upper left or right corner.

**Note:** Depends upon the position setting. For example, if Left-Align is chosen, the value in X refers to the selection list upper left corner.

### Y

Enter a value in pixels from the window top edge where you want to place the selection list upper left or right corner.

**Note:** Depends upon the position setting. For example, if Left-Align is chosen, the value in Y refers to the selection list upper left corner.

### Width Pixels

Enter a value for the number of pixel units wide to make the selection list.

### Height Pixels

Enter a value for the number of pixel units high to make the selection list.

### Left-Align

Select this option to position the selection list left edge along the column / X position specified.

### Right-Align

Select this option to position the selection list right edge along the column / X position specified.

### Advanced Settings

This section allows you to select advanced run-time attributes and other ABL related options.

### Current Layout

Displays the current layout used to display the selection list.

### Sync With Master

Displays the Sync With Master dialog which allows you to overwrite the current layout with the selection list default properties.

**Note:** The Sync With Master option is enabled if you have defined a master layout.

---

For more information, see Property Sheet dialogs on page 702
## Advanced properties for Editor

The **Advanced Properties** dialog allows you to set advanced properties for the selected editor.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Initial Value</th>
<th>Enter an initial value for the editor. The GUI Designer writes this value to the DEFINE VARIABLE statement for the editor.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Help</td>
<td>Enter a help string for the editor. The GUI Designer displays the help string in the run-time window status bar whenever the editor has input focus. By default the AppBuilder uses help you define here unless you are using a database variable that has help defined in the schema.</td>
</tr>
<tr>
<td>Private Data</td>
<td>Enter any private data about the editor. Private data is an annotation containing any character data that you can attach to an object. This annotation is only informational and not used by the code. The GUI Designer reads and writes private data as comments in the Run-time Attribute and AppBuilder Settings section of procedure files.</td>
</tr>
<tr>
<td>Characters</td>
<td>Select this option to write editor size and position in character units. The GUI Designer writes the values entered in Column, Row, Width, and Height on the EDITOR Property Sheet in character units using the Size and At phrases in the DEFINE VARIABLE and DEFINE FRAME statements.</td>
</tr>
<tr>
<td>Pixels</td>
<td>Select this option to express editor size and position in pixel units. The GUI Designer writes the values entered in X, Y, Width Pixels, and Height Pixels in pixel units using the Size and At phrases in the DEFINE FRAME statement.</td>
</tr>
</tbody>
</table>
| {&List-1}     | Select this option to add the editor name to the selected custom list. For example, select the {&List-1} check box to add the editor name to custom list &List-2. One of six user-defined, alphabetical lists of space-delimited object names represented by the preprocessor names (&List-1, &List-2, &List-3, &List-4, &List-5, &List-6). You can use custom lists in ABL code whenever you need to represent object names, as in the following example:  

```
ENABLE ALL EXCEPT {&LIST-1} WITH FRAME {&FRAME-NAME}.
HIDE {&LIST-2} WITH FRAME {&FRAME-NAME}.
```

The GUI Designer adds an object name to a custom list whenever you activate one of the list toggles on the object's Advanced Property Sheet. Objects that are cut or deleted from a procedure file are automatically removed from the appropriate list. |
| X             | Enter a value in pixels from the window left edge where you want to place the editor upper left or right corner. |

**Note:** Depends upon the position setting. For example, if Left-Align is chosen, the value in X refers to the editor upper left corner.
Enter a value in pixels from the window top edge where you want to place the editor upper left or right corner.

**Note:** Depends upon the position setting. For example, if Left-Align is chosen, the value in Y refers to the editor upper left corner.

<table>
<thead>
<tr>
<th>Width Pixels</th>
<th>Enter a value for the number of pixel units wide to make the editor.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Height Pixels</td>
<td>Enter a value for the number of pixel units high to make the editor.</td>
</tr>
</tbody>
</table>

Select this option to position the editor label left edge along the column / X position specified.

Select this option to position the editor right edge along the column / X position specified.

This section allows you to select advanced run-time attributes and other ABL related options.

Displays the current layout used to display the editor.

Displays the **Sync With Master** dialog which allows you to overwrite the current layout with the editor default properties.

**Note:** The **Sync With Master** option is enabled if you have defined a master layout.

For more information, see Property Sheet dialogs on page 702

### Advanced properties for Combo Box

The **Advanced Properties** dialog allows you to set advanced properties for the selected combo box.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Initial Value</th>
<th>Enter an initial value for the combo box. The GUI Designer writes this value to the DEFINE VARIABLE statement for the combo box.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Help</td>
<td>Enter a help string for the combo box. The GUI Designer displays the help string in the run-time window status bar whenever the combo box has input focus. By default the AppBuilder uses help you define here unless you are using a database variable that has help defined in the schema.</td>
</tr>
<tr>
<td>Private Data</td>
<td>Enter any private data about the combo box. Private data is an annotation containing any character data that you can attach to an object. This annotation is only informational and not used by the code. The GUI Designer reads and writes private data as comments in the Run-time Attribute and AppBuilder Settings section of procedure files.</td>
</tr>
<tr>
<td>Characters</td>
<td>Choose to write combo box size and position in character units. The AppBuilder writes the values entered in Column, Row, Width, and Height on the COMBO-BOX Property Sheet in character units using the Size and At phrases in the DEFINE VARIABLE and DEFINE FRAME statements.</td>
</tr>
</tbody>
</table>
Choose to express combo box size and position in pixel units. The AppBuilder writes the values entered in X, Y, Width Pixels, and Height Pixels in pixel units using the Size and At phrases in the DEFINE VARIABLE and DEFINE FRAME statements.

Select this option to add the combo box name to the selected custom list. For example, select the {&List-1} checkbox to add the combo box name to custom list &List-2.

One of six user-defined, alphabetical lists of space-delimited object names represented by the preprocessor names (&List-1, &List-2, &List-3, &List-4, &List-5, &List-6). You can use custom lists in ABL code whenever you need to represent object names, as in the following example:

- **Enable All Except**
  - ENABLE ALL EXCEPT &List-1 WITH FRAME &Frame-Name.
  
- **Hide**
  - HIDE &List-2 WITH FRAME &Frame-Name.

The GUI Designer adds an object name to a custom list whenever you activate one of the list toggles on the object's Advanced Property Sheet. Objects that are cut or deleted from a procedure file are automatically removed from the appropriate list.

Enter a value in pixels from the window left edge where you want to place the combo box upper left or right corner.

**Note:** Depends upon the position setting. For example, if Left-Align is chosen, the value in X refers to the combo box upper left corner.

Enter a value in pixels from the window top edge where you want to place the combo box upper left or right corner.

**Note:** Depends upon the position setting. For example, if Left-Align is chosen, the value in Y refers to the combo box upper left corner.

Enter a value for the number of pixel units wide to make the combo box.

Enter a value for the number of pixel units high to make the combo box.

Select this option to position the fill-in left edge (or label left edge) along the column / X position specified.

Select this option to position the fill-in label colon (:) along the column / X position specified. (If there is no label, then the AppBuilder aligns the fill-in as if it had a label.)

Select this option to position the fill-in right edge along the column position / X specified.

This section allows you to select advanced run-time attributes and other ABL related options.
Current Layout | Displays the current layout used to display the combo box.
---|---
Sync With Master | Displays the Sync With Master dialog which allows you to overwrite the current layout with the combo box default properties.

**Note:** The Sync With Master option is enabled if you have defined a master layout.

For more information, see Property Sheet dialogs on page 702

**Advanced properties for Fill In**

The Advanced Properties dialog allows you to set advanced properties for the selected fill-in.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Initial Value</th>
<th>Enter an initial value for the fill-in. The GUI Designer writes this value to the DEFINE VARIABLE statement for the fill-in.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Help</td>
<td>Enter a help string for the fill-in. The GUI Designer displays the help string in the run-time window status bar whenever the fill-in has input focus. By default the AppBuilder uses help you define here unless you are using a database variable that has help defined in the schema.</td>
</tr>
<tr>
<td>Private Data</td>
<td>Enter any private data about the fill-in. Private data is an annotation containing any character data that you can attach to an object. This annotation is only informational and not used by the code. The GUI Designer reads and writes private data as comments in the Run-time Attribute and AppBuilder Settings section of procedure files.</td>
</tr>
<tr>
<td>Characters</td>
<td>Select this option to write fill-in size and position in character units. The GUI Designer writes the values entered in Column, Row, Width, and Height on the EDITOR Property Sheet in character units using the Size and At phrases in the DEFINE VARIABLE and DEFINE FRAME statements.</td>
</tr>
<tr>
<td>Pixels</td>
<td>Select this option to express fill-in size and position in pixel units. The GUI Designer writes the values entered in X, Y, Width Pixels, and Height Pixels in pixel units using the Size and At phrases in the DEFINE FRAME statement.</td>
</tr>
</tbody>
</table>
| Custom Lists | Select this option to add the fill-in name to the selected custom list. For example, select the {&List-1} check box to add the fill-in name to custom list &List-2. One of six user-defined, alphabetical lists of space-delimited object names represented by the preprocessor names (&List-1, &List-2, &List-3, &List-4, &List-5, &List-6). You can use custom lists in ABL code whenever you need to represent object names, as in the following example:

```
ENABLE ALL EXCEPT {&LIST-1} WITH FRAME {&FRAME-NAME}.
HIDE {&LIST-2} WITH FRAME {&FRAME-NAME}.
```

The GUI Designer adds an object name to a custom list whenever you activate one of the list toggles on the object's Advanced Property Sheet. Objects that are cut or deleted from a procedure file are automatically removed from the appropriate list.
X  Enter a value in pixels from the window left edge where you want to place the fill-in upper left or right corner.  

**Note:** Depends upon the position setting and whether the fill-in has a label. For example, if Left-Align is chosen, the value in X refers to the fill-in upper left corner.

Y  Enter a value in pixels from the window top edge where you want to place the fill-in upper left or right corner.  

**Note:** Depends upon the position setting. For example, if Left-Align is chosen, the value in Y refers to the fill-in upper left corner.

Width Pixels  Enter a value for the number of pixel units wide to make the fill-in.

Height Pixels  Enter a value for the number of pixel units high to make the fill-in.

Left-Align  Select this option to position the fill-in left edge (or label left edge) along the column / X position specified.

Colon-Align  Select this option to position the fill-in label colon (:) along the column / X position specified. (If there is no label, then the AppBuilder aligns the fill-in as if it had a label.)

Right-Align  Select this option to position the fill-in right edge along the column position / X specified.

Advanced Settings  This section allows you to select advanced run-time attributes and other ABL related options.

Current Layout  Displays the current layout used to display the fill-in.

Sync With Master  Displays the **Sync With Master** dialog which allows you to overwrite the current layout with the fill-in default properties.  

**Note:** The **Sync With Master** option is enabled if you have defined a master layout.

For more information, see [Property Sheet dialogs on page 702](#).

**Advanced properties for Text**

The **Advanced Properties** dialog allows you to to set advanced properties for the selected text object. This dialog includes the following options:

<table>
<thead>
<tr>
<th>Characters</th>
<th>Select this option to write text object size and position in character units. The GUI Designer writes the values entered in Column, Row, Width, and Height on the TEXT Property Sheet in character units using the Size and At phrases in the DEFINE FRAME statement.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pixels</td>
<td>Select this option to write text object size and position in pixel units. The GUI Designer writes the values entered in X, Y, Width Pixels, and Height Pixels in pixel units using the Size and At phrases in the DEFINE FRAME statement.</td>
</tr>
</tbody>
</table>
**Chapter 14: Introducing the ABL GUI Designer**

<table>
<thead>
<tr>
<th>X</th>
<th>Enter a value in pixels from the window left edge where you want to place the text object upper left or right corner.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Note:</strong> Depends upon the position setting. For example, if Left-Align is chosen, the value in X refers to the text object upper left corner.</td>
</tr>
<tr>
<td>Y</td>
<td>Enter a value in pixels from the window top edge where you want to place the text object upper left or right corner.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Depends upon the position setting. For example, if Left-Align is chosen, the value in Y refers to the text object upper left corner.)</td>
</tr>
</tbody>
</table>

| Width Pixels | Enter a value for the number of pixel units wide to make the text object. |
| Height Pixels | Enter a value for the number of pixel units high to make the text object. |
| Left-Align | Select this option to position the text object left edge along the column / X position specified. |
| Right-Align | Select this option to position the text object right edge along the column / X position specified. |
| Current Layout | Displays the current layout used to display the text object. |

For more information see, *Property Sheet dialogs* on page 702

**Advanced properties for OCX**

The **Advanced Properties** dialog allows you to set advanced properties for the selected OCX control frame. This dialog includes the following options:

<table>
<thead>
<tr>
<th>Help</th>
<th>Enter a help string for the OCX control frame. The AppBuilder displays the help string in the run-time window status bar whenever the button has input focus.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Private Data</td>
<td>Enter any private data about the OCX control frame.</td>
</tr>
<tr>
<td>Characters</td>
<td>Select this option to write the control frame size and position in character units. The GUI Designer writes the values entered in Column, Row, Width, and Height on the OCX Control Frame Property Sheet in character units in the CREATE CONTROL-FRAME statement.</td>
</tr>
<tr>
<td>Pixels</td>
<td>Select this option to write OCX control frame size and position in pixel units. The GUI Designer writes the values entered in X, Y, Width Pixels, and Height Pixels in pixel units in the CREATE CONTROL-FRAME statement.</td>
</tr>
</tbody>
</table>
Custom Lists | Select this option to add the OCX control frame name to the selected custom list. For example, select the \{&List-1\} check box to add the OCX control frame name to custom list &List-2.

One of six user-defined, alphabetical lists of space-delimited object names represented by the preprocessor names \{&List-1, &List-2, &List-3, &List-4, &List-5, &List-6\}. You can use custom lists in ABL code whenever you need to represent object names, as in the following example:

ENABLE ALL EXCEPT \{&LIST-1\} WITH FRAME \{&FRAME-NAME\}.
HIDE \{&LIST-2\} WITH FRAME \{&FRAME-NAME\}.

The GUI Designer adds an object name to a custom list whenever you activate one of the list toggles on the object's Advanced Property Sheet. Objects that are cut or deleted from a procedure file are automatically removed from the appropriate list.

<table>
<thead>
<tr>
<th>X</th>
<th>Enter a value in pixels from the window left edge where you want to place the control frame upper left or corner.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>Enter a value in pixels from the window top edge where you want to place the OCX control frame upper left corner.</td>
</tr>
<tr>
<td>Width Pixels</td>
<td>Enter a value for the number of pixel units wide to make the OCX control frame.</td>
</tr>
<tr>
<td>Height Pixels</td>
<td>Enter a value for the number of pixel units high to make the OCX control frame height.</td>
</tr>
<tr>
<td>Current Layout</td>
<td>Displays the current layout used to display the OCX control frame.</td>
</tr>
<tr>
<td>Sync With Master</td>
<td>Displays the Sync With Master dialog which allows you to overwrite the current layout with the OCX control frame default properties.</td>
</tr>
</tbody>
</table>

**Note:** The Sync With Master option is enabled if you have defined a master layout.

For more information, see Property Sheet dialogs on page 702

### Choose Color dialog

The Choose Color dialog helps you to customize the background and foreground colors of objects and frame in the design window.

The color numbers displayed are logical representations of either OpenEdge reserved RGB values or customized RGB values. Colors zero through 15 are OpenEdge reserved colors. You can customize only colors 16 through 255.

You can access this dialog by clicking on the Color icon on the object's property sheet dialog or Column Editor dialog (for Browse) on page 769.

The Choose Color dialog includes the following options:

<table>
<thead>
<tr>
<th>Sample</th>
<th>Displays an example of currently selected background and foreground colors.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreground Colors</td>
<td>Select a foreground color. The foreground color you select appears in the Sample field. The question mark (?) represents the default color for the object, which is determined by its parent.</td>
</tr>
</tbody>
</table>
Select a background color. The background color you select appears in the Sample field. The question mark (?) represents the default color for the object, which is determined by its parent.

- Displays the previous sixteen colors in the palette.
- Displays the next sixteen colors in the palette.
- Displays the Color dialog that allows you to edit the selected foreground or background color.
- Select to save your current color selections for future sessions.

For more information about color and font inheritance, see the *OpenEdge Development: ABL Handbook*.

**See also**

- Changing color of an object on page 650

**Character Terminal Simulator Colors dialog**

The *Character Terminal Simulator Colors* dialog allows you to customize the background and foreground colors for your OpenEdge session. The color numbers displayed are logical representations of either OpenEdge reserved RGB values or customized RGB values.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Sample</th>
<th>Displays an example of currently selected background and foreground colors.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreground Colors</td>
<td>Select a foreground color. The foreground color you select appears in the Sample field. The question mark (?) represents the default color for the object, which is determined by its parent.</td>
</tr>
<tr>
<td>Background Colors</td>
<td>Select a background color. The background color you select appears in the Sample field. The question mark (?) represents the default color for the object, which is determined by its parent.</td>
</tr>
<tr>
<td>Displays the previous sixteen colors in the palette.</td>
<td></td>
</tr>
<tr>
<td>Displays the next sixteen colors in the palette.</td>
<td></td>
</tr>
<tr>
<td>Edit</td>
<td>Displays the Color dialog that allows you to select and edit color. Colors zero through 15 are OpenEdge reserved colors. You can customize only colors 16 through 255.</td>
</tr>
<tr>
<td>Save Color Settings</td>
<td>Select to save your current color selections for future sessions.</td>
</tr>
</tbody>
</table>

For more information, see *AppBuilder properties* page.

**Choose Font dialog**

The *Choose Font* dialog helps you to choose or customize the fonts for your OpenEdge session.
The font numbers displayed are logical representations of either OpenEdge reserved system fonts or customized system fonts.

You can access this dialog by clicking on the Font icon on the object's property sheet dialog or Column Editor dialog (for Browse) on page 769.

The Choose Font dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample</td>
<td>Displays an example of currently selected font.</td>
</tr>
<tr>
<td></td>
<td>Select to display the previous four available fonts.</td>
</tr>
<tr>
<td></td>
<td>Select to display the next four available fonts.</td>
</tr>
<tr>
<td>Edit</td>
<td>Opens the Font dialog that allows you to edit the selected font.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Font zero through seven are OpenEdge reserved fonts. You can customize only fonts eight through 255.</td>
</tr>
<tr>
<td>Save Color</td>
<td>Select to save your current font selections for future sessions.</td>
</tr>
<tr>
<td>Settings</td>
<td>OK</td>
</tr>
<tr>
<td>Cancel</td>
<td>Select to ignore your changes and exit the dialog.</td>
</tr>
</tbody>
</table>

See also

- Column Editor dialog (for Browse) on page 769
- Choose Color dialog on page 753

Choose Control dialog

The Choose Control dialog allows you to insert an OCX control on the design window.

Access this dialog box by selecting OCX from the OCX section on the Palette.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available Controls</td>
<td>List of the registered ActiveX controls available on your system, and the three controls provided with OpenEdge: CSComboBox, CSSpin, and PSTimer.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> You can also access the three controls provided with OpenEdge directly from the GUI Designer Palette.</td>
</tr>
<tr>
<td></td>
<td>When you select a control, a preview of the control appears below the Available Controls list along with its corresponding complete path and filename.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Browse</td>
<td>Open the browse dialog to search for a control by filename from other directories on your system.</td>
</tr>
<tr>
<td>OK</td>
<td>Insert the selected OCX control on the design window.</td>
</tr>
</tbody>
</table>

For more information, see [ActiveX objects](#) on page 684

**Choose Template dialog**

The Choose Other Template dialog allows you to select GUI Designer procedure file as a template and place it in the objects list for the current GUI Designer session. You can use the template to create an ABL GUI procedure file.

Access this dialog by clicking Template on the Create an ABL GUI procedure wizard.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>File</th>
<th>Enter a file name, or select a file from files list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>File List</td>
<td>Lists the files from the directory specified in the Directory and File Type drop-down.</td>
</tr>
<tr>
<td>File Type</td>
<td>Select a type of file to display in the files list.</td>
</tr>
<tr>
<td>Directory</td>
<td>Specify the directory to search for files. All the files present in the selected directory appears in the files list.</td>
</tr>
<tr>
<td>Source</td>
<td>If the Preview check box is selected, displays the contents of the selected file from the files list. This is a read-only editor.</td>
</tr>
<tr>
<td>Preview</td>
<td>Select this check box to display the contents of the selected file in the source editor.</td>
</tr>
<tr>
<td>Code/Image</td>
<td>When preview is selected, this radio set indicates the type of file being previewed.</td>
</tr>
<tr>
<td>Edit Path</td>
<td>Displays the Edit Path dialog which allows you to add or remove the source directories.</td>
</tr>
<tr>
<td>Browse</td>
<td>Displays the Browse Files dialog which allows you to search entire system for directories and files.</td>
</tr>
</tbody>
</table>

For more information, see [Creating ABL GUI procedure files](#) on page 644

**Table Selector dialog**

The Table Selector dialog allows you to select a table for the operation you are performing.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Databases</th>
<th>Lists all the connected databases, and allows you to select a database from the list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tables</td>
<td>Lists all the tables of the selected database, and allows you to select a table.</td>
</tr>
</tbody>
</table>

For more information, see [Multi-Field Selector dialog](#) on page 757
Field Selector dialog box

Use the Field Selector dialog box to choose fields for the operation you are performing.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Databases</td>
<td>Choose a database. Displays all the connected databases.</td>
</tr>
<tr>
<td>Tables</td>
<td>Choose a table. Displays all the tables of the selected database.</td>
</tr>
<tr>
<td>Fields</td>
<td>Choose fields. You can choose more than one field by pressing and holding CTRL while you select the field names with the mouse. Displays all the fields of the selected table. (Fields only displays when a table is selected.)</td>
</tr>
</tbody>
</table>

See also

Property Sheet for Slider on page 719

Multi-Field Selector dialog

The Multi-Field Selector dialog allows you to select fields for the operation you are performing.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Available Fields</th>
<th>Lists the fields of the table selected from the Table Selector dialog. Highlight one or more fields to add to the Selected Fields list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selected Fields</td>
<td>Lists the fields that you add from the Available Fields list. Highlight one or more fields to remove, move up, or move down in this list.</td>
</tr>
<tr>
<td>Add &gt;&gt;</td>
<td>Add fields to the Selected Fields list. Moves the highlighted fields in the Available Fields list (left list) to the Selected Fields list (right list).</td>
</tr>
<tr>
<td>&lt;&lt; Remove</td>
<td>Removes fields from the Selected Fields list. Moves the highlighted fields in the Selected Fields list (right list) to the Available Fields list (left list)</td>
</tr>
<tr>
<td>Move UP</td>
<td>Moves the highlighted fields in the Selected Fields list (right list) one position up in the list.</td>
</tr>
<tr>
<td>Move Down</td>
<td>Moves the highlighted fields in the Selected Fields list (right list) one position down in the list.</td>
</tr>
</tbody>
</table>

For more information, see Table Selector dialog on page 756

Add Trigger dialog

The Add Trigger dialog helps you associate an event (also called a trigger) with an object in the AppBuilder (.w) file. This option is available only with the AppBuilder (.w) files.
You can access this wizard from the **Source menu > Add Trigger** or right-click on the ABL Editor and select **Source > Add Trigger** from the context menu.

The **Add Trigger** wizard includes the following controls:

<table>
<thead>
<tr>
<th>Widget</th>
<th>Select an object from the drop-down to which you want to add trigger. This list contains all the objects that are present on the current design window.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event category</td>
<td>Select an event category from the drop-down. This lists contains the following event categories:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Common Events</strong> – Select this to display all common events for the selected object.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Direct Manipulation Events</strong> – Select this to display the direct manipulation events. These are explicit actions a user performs on objects with a keyboard or a mouse such as selecting, highlighting, moving, and resizing.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Portable Mouse Events</strong> – Select this to display the portable mouse events. These are mouse events based on a conceptual four-button mouse model with select, menu, move, and extend buttons.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Three-Button Mouse Events</strong> – Select this to display the three-button mouse events. These are mouse events based on three-button mouse model with left, middle, and right buttons.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Developer Events</strong> – Select this to display the developer events. There are ten system-supplied events (U1 to U10) that you can apply to objects with the <strong>APPLY</strong> statement.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Keyboard Events</strong> – Select this to record keystrokes and bind them to events. When you select this option <strong>Key label</strong>, <strong>Key function</strong>, and <strong>Generate code</strong> by options will be available and grouped under <strong>Event name</strong> section.</td>
</tr>
<tr>
<td>Event name</td>
<td>Select an event name from the drop-down. This list contains the event names based on the selection you make in the <strong>Widget</strong> and <strong>Event category</strong> drop-downs.</td>
</tr>
<tr>
<td>Define key (Key label)</td>
<td>Enter a label for the keyboard event such as <strong>F1</strong> or <strong>F2</strong>.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: This option is available when you specify Keyboard Events in the <strong>Event category</strong> drop-down.</td>
</tr>
<tr>
<td>Key function</td>
<td>Displays the pre-defined function name associated with the key label you have entered in the <strong>Key label</strong> field such as <strong>Help</strong> for <strong>F1</strong> or <strong>GO</strong> for <strong>F2</strong>. If there is no pre-defined function associated with the specified key label, then the same key label text is displayed in the <strong>Key function</strong> field.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: This option is available when you specify Keyboard Events in the <strong>Event category</strong> drop-down.</td>
</tr>
<tr>
<td>Generate code by</td>
<td>Allows you to generate the trigger code by either the <strong>Key label</strong> or <strong>Key function</strong>.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: This option is available when you specify Keyboard Events in the <strong>Event category</strong> drop-down.</td>
</tr>
</tbody>
</table>
Generate the trigger code for the selected object and inserts it in the (.w) procedure file.

Select to ignore your changes and exit the trigger wizard.

**See also**
Connecting to a database on page 655

**New ADM Class dialog**

The **New ADM Class** dialog allows you to create a new ADM2Class.

ADM class is a collection of files that define properties and run-time routines for a particular SmartObject or family of SmartObjects.

You can access this dialog from the **File** menu, by choosing **New ADM Class**.

---

**Note:** The first step in creating a new ADM2Class is to specify a class name in the **Name** field. Once you provide a class name, all of the file names are automatically generated. This is done to provide (or at least to suggest) a standard file–naming convention that is consistent with the standard ADM classes. The files can then be renamed if needed, but once the new class is saved they can no longer be changed.

The **Basic** tab on the **New ADM Class** dialog includes the following fields:

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Specify a name for the ADM 2 class. This name serves as a base name for the name of all the files. This is a mandatory field.</td>
</tr>
<tr>
<td>Class Definition File</td>
<td>Specify a file that references the components of a class. The extension must be <code>.cld</code>. This is a mandatory field.</td>
</tr>
<tr>
<td>Source Directory</td>
<td>Specify the path where the source files will be generated. Click <strong>Browse</strong> to select a different path. This is a mandatory field.</td>
</tr>
<tr>
<td>Rcode Directory</td>
<td>Specify the path where the r-code for the super procedure is generated. Click <strong>Browse</strong> to select a different path. This is a mandatory field.</td>
</tr>
<tr>
<td>Template Directory</td>
<td>Specify the location of the template file. Click <strong>Browse</strong> to select a different location. This is a mandatory field, if a template file name is provided in the <strong>Template</strong> field.</td>
</tr>
<tr>
<td>Derive From Class</td>
<td>Class definition file (.cld) to subclass. Click <strong>Class</strong> to browse and select a class definition file.</td>
</tr>
</tbody>
</table>
### Field name | Description
---|---
Method Library | Specify a file that defines the class name, references a property file, and starts the super procedure. The extension must be .i. This is a mandatory field.

Property File | Specify a file that defines class properties. The extension must be .i. This is a mandatory field.

Super Procedure | Specify a file that defines get/set functions for readable/writable properties. It defines new behavior for the class. The extension must be .p. This is a mandatory field.

Prototype File | Specify a file that references functions and internal procedures of a super procedure. The extension must be .i. The contents of this file can be generated by using the ProtoGen tool from the PRO*Tools toolbar. This is a mandatory field.

Template | Specify a name of the template file. References the method library of the class.

Copy From Template | Specify a template file to copy from. Click **File** to browse and select a template file.

Open files in the AppBuilder once generated | Select this option to open the generated files in AppBuilder.

The **Custom Files** tab on the **New ADM Class** dialog lists the files to be generated to allow customization of the new class you have created. The fields available on this tab are read-only fields.

For more information, see [AppBuilder dialogs on page 699](#).

### Progress Advisor dialog

You use the **Freeform Query** option to build queries using temporary tables, extra buffers, and preprocessor statements. Once you choose the **Freeform** option, you cannot return to using the Query Builder to maintain freeform queries, and can only maintain the query (and browse fields if the query is for a BROWSE object) through the Source view or Source Editor.

Choose from the radio set whether to change the selected query to a freeform query or cancel the operation.

### Query Builder dialogs

**Query Builder Dialog Box -- Join Mode**

The **Query Builder** dialog in the Join mode allows you to create custom joins in ABL for the selected query. Access this dialog by selecting **Join** option on the **Query Builder** dialog.

This dialog includes the following options:

<p>| Join | Specify which join in the query being built to customize. Select the query from the Join list. |</p>
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fields</strong></td>
<td>Displays the Column Editor dialog which allows you to specify which fields to display in the browse on the design canvas. <strong>Note:</strong> This option is only available when editing a query associated with a browse widget.</td>
</tr>
<tr>
<td>(table)</td>
<td>Displays all the fields for table on the left of the selected join. Use to add fields to the Join Criteria editor.</td>
</tr>
<tr>
<td>(table)</td>
<td>Displays all the fields for the table on the right of the selected join. Use to add fields to the Join Criteria editor.</td>
</tr>
<tr>
<td>= (Equal)</td>
<td>Select this option to construct an equal (=) comparison between fields. This generates the comparison in ABL code in the Join Criteria editor.</td>
</tr>
<tr>
<td>&lt;&gt; (Not Equal)</td>
<td>Select this option to construct a not equal (&lt;&gt;) comparison between fields. This generates the comparison in ABL code in the Join Criteria editor.</td>
</tr>
<tr>
<td>&lt; (Less Than)</td>
<td>Select this option to construct a less than (&lt;) comparison between fields. This generates the comparison in ABL code in the Join Criteria editor.</td>
</tr>
<tr>
<td>&gt; (Greater Than)</td>
<td>Select this option to construct a greater than (&gt;) comparison between fields. This generates the comparison in ABL code in the Join Criteria editor.</td>
</tr>
<tr>
<td>&lt;= (Less Than or Equal)</td>
<td>Select this option to construct a less than or equal (&lt;=) comparison between fields. This generates the comparison in ABL code in the Join Criteria editor.</td>
</tr>
<tr>
<td>&gt;= (Greater Than or Equal)</td>
<td>Select this option to construct a greater than or equal (&gt;=) comparison between fields. This generates the comparison in ABL code in the Join Criteria editor.</td>
</tr>
<tr>
<td><strong>AND</strong></td>
<td>Select this option to construct an AND conjunction between two logical expressions in the ABL code in the Join Criteria editor.</td>
</tr>
<tr>
<td><strong>OR</strong></td>
<td>Select this option to construct an OR conjunction between two logical expressions in the ABL code in the Join Criteria editor.</td>
</tr>
<tr>
<td><strong>Customize Join</strong></td>
<td>Select this option to change the default join criteria using the WHERE clause. By default, the GUI Designer attempts to join tables using OF.</td>
</tr>
<tr>
<td><strong>Join Criteria</strong></td>
<td>Displays the join you are creating. You may also edit the query directly in this editor when it has focus.</td>
</tr>
</tbody>
</table>
Select this option to undo the last edit made to the Join Criteria editor.

Select this option to have the compiler check the syntax of the query in its current state.

Select this option to have the compiler check the syntax of the query when you click OK.

Displays the PROGRESS Advisor dialog which allows freeform editing of the query using the GUI Designer editor.

See also
QueryBuilder Dialog Box -- Table Mode on page 764
QueryBuilder Dialog Box -- Where Mode on page 765
QueryBuilder Dialog Box -- Sort Mode on page 763
QueryBuilder Dialog Box -- Options Mode on page 762

Query Builder Dialog Box -- Options Mode

The Query Builder dialog in Options mode allows you to modify the query attributes listed below and query tuning parameters for the selected query.

Note: Query tuning parameters are used to fine-tune query performance. This option is available only with non-OpenEdge databases. Please refer to your data server documentation for more information on this topic.

Access this mode by selecting Options on the Query Builder dialog.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Fields</th>
<th>Displays the Column Editor dialog which allows you to specify which fields to display in the browse.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>This option is only available when editing a query associated with a browse widget.</td>
</tr>
<tr>
<td>Table</td>
<td>Lists the current tables in the selected query.</td>
</tr>
<tr>
<td>Find</td>
<td>Lists the find parameter for table in the selected row. Use to set the find parameter for a table to FIND EACH, FIND FIRST, or FIND LAST. The default is FIND EACH. You can double-click in this field to cycle through the valid choices or press 'E', 'F' or 'L' to select Find Each, Find First or Find Last, respectively.</td>
</tr>
<tr>
<td>Join</td>
<td>Lists whether the joins for the tables in the selected row are inner joins or outer joins. Use to set the join type for a table to inner or outer. The default join type is inner. You can double-click to cycle through the valid choices or press 'I' for Inner or 'O' for Outer.</td>
</tr>
</tbody>
</table>
Lists what fields the selected query will return for each table. Use to set the fields in the Field-List for a table to All Fields or Fields Used. When you select Fields Used, the query returns only those fields in the FIELDS-IN-QUERY preprocessor list. The default is All Fields. You can double-click to cycle through the valid choices or press 'A' for All Fields or 'F' for Fields Used.

Enter ABL code in this editor to specify query tuning parameters for the selected query. It is recommended to reference an include file so that all queries in an application can be easily modified by modifying the contents of the include file instead of referencing this dialog for each query.

Select this check box to add the Key-Phrase preprocessor variable to the query definition.

Select this check box to add the SortBy-Phrase preprocessor variable to the query definition. If you use select this option, you cannot directly modify the Sort from the Query Builder.

Select this option to have the compiler check the syntax of the query in its current state.

Select this option to have the compiler check the query syntax when you click OK.

Displays the PROGRESS Advisor dialog which allows freeform editing of the query using the GUI Designer editor.

---

**See also**

*QueryBuilder Dialog Box -- Join Mode* on page 760
*QueryBuilder Dialog Box -- Where Mode* on page 765
*QueryBuilder Dialog Box -- Sort Mode* on page 763
*QueryBuilder Dialog Box -- Table Mode* on page 764

---

**Query Builder Dialog Box -- Sort Mode**

The *Query Builder* dialog in the Sort mode allows you to generate a SORT clause in ABL for the selected query.

Access this mode by selecting the *Sort* option on the *Query Builder* dialog.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Table</th>
<th>Select table for the sort clause from the list of available tables.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fields</td>
<td>Displays the Column Editor dialog, which allows you to specify which fields to display in the browse on the design canvas.</td>
</tr>
</tbody>
</table>

**Note:** This option is only available when editing a query associated with a browse widget.
<table>
<thead>
<tr>
<th>Available Fields</th>
<th>Displays all the fields for the selected table. Select a field with which to specify sort criteria.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selected Fields</td>
<td>Displays the fields selected to include in the sort criteria.</td>
</tr>
<tr>
<td>Add &gt;&gt;</td>
<td>Select this option to add fields to the <strong>Selected Fields</strong> list. Moves the highlighted fields in the <strong>Available Fields</strong> list to the <strong>Selected Fields</strong> list.</td>
</tr>
<tr>
<td>&lt;&lt; Remove</td>
<td>Select this option to remove fields from the <strong>Selected Fields</strong> list. Moves the highlighted fields in the <strong>Selected Fields</strong> list to the <strong>Available Fields</strong> list.</td>
</tr>
<tr>
<td>Move Up</td>
<td>Select this option to move the highlighted fields one position in the <strong>Selected Fields</strong> list.</td>
</tr>
<tr>
<td>Move Down</td>
<td>Select this option to move the highlighted field one position down in the <strong>Selected Fields</strong> list.</td>
</tr>
<tr>
<td>Ascending</td>
<td>Select this option to sort the fields in ascending order. This is the default sort option.</td>
</tr>
<tr>
<td>Descending</td>
<td>Select this option to sort the fields in descending order.</td>
</tr>
<tr>
<td>Sort Criteria</td>
<td>Displays the sort criteria you are creating.</td>
</tr>
<tr>
<td>Freeform Query</td>
<td>Displays the <strong>PROGRESS Advisor</strong> dialog which allows freeform editing of the query using the GUI Designer editor.</td>
</tr>
</tbody>
</table>

**Query Builder Dialog Box -- Table Mode**

The **Query Builder** dialog in the Table mode allows you to add and remove tables to the selected query.

Access this mode by selecting the **Table** option on the **Query Builder** dialog.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Database</th>
<th>Select a database from the list of connected databases.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fields</td>
<td>Displays the <strong>Column Editor</strong> dialog which allows you to specify which fields to display in the browse on the design canvas.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This option is only available when editing a query associated with a browse widget.</td>
</tr>
<tr>
<td>Available Tables</td>
<td>Displays the tables of the selected database. You can select tables to add to the <strong>Selected Tables &amp; Joins</strong> section.</td>
</tr>
<tr>
<td>Add &gt;&gt;</td>
<td>Allows you to add selected tables to the <strong>Selected Tables &amp; Joins</strong> list. These tables are included in the query that the GUI Designer creates.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>&lt;&lt; Remove</td>
<td>Allows you to remove tables from the <strong>Selected Tables &amp; Joins</strong> list.</td>
</tr>
<tr>
<td>Selected Tables &amp; Joins</td>
<td>Displays the tables that are selected from the <strong>Available Tables</strong> with their join partners that you have created.</td>
</tr>
<tr>
<td>Lock-Type</td>
<td>Allows you to select the table access mode for the defined query. The available options are: No-lock (the default), Share-lock, or Exclusive-lock.</td>
</tr>
<tr>
<td>Switch Join Partners</td>
<td>Select this option to change the table in the highlighted join.</td>
</tr>
<tr>
<td>Query</td>
<td>Displays the query you are creating.</td>
</tr>
<tr>
<td>Indexed-Reposition</td>
<td>Select this option to enable INDEXED-REPOSITION. If you specify this option, OpenEdge attempts to optimize subsequent REPOSITION TO RECID operations on the query. (Only available for single table queries.)</td>
</tr>
<tr>
<td>Check Syntax Now</td>
<td>Select this option to have the compiler check the syntax of the query in its current state.</td>
</tr>
<tr>
<td>Check Syntax On OK</td>
<td>Select this option to have the compiler check the syntax of the query when you click OK.</td>
</tr>
<tr>
<td>Freeform Query</td>
<td>Displays the <strong>PROGRESS Advisor</strong> dialog which allows freeform editing of the query using the GUI Designer editor.</td>
</tr>
</tbody>
</table>

**See also**
- Query Builder Dialog Box -- Join Mode on page 760
- Query Builder Dialog Box -- Where Mode on page 765
- Query Builder Dialog Box -- Sort Mode on page 763
- Query Builder Dialog Box -- Options Mode on page 762

**Query Builder Dialog Box -- Where Mode**

The **Query Builder** dialog in the Where mode allows you to generate a WHERE clause to specify search criteria in ABL for the selected query.

Access this mode by selecting the Where option on the **Query Builder** dialog.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>Select table for the where clause from the list of available tables.</td>
</tr>
<tr>
<td>Fields</td>
<td>Displays the Column Editor dialog, which allows you to specify which fields to display in the browse on the design canvas.</td>
</tr>
</tbody>
</table>

**Note:** This option is only available when editing a query associated with a browse widget.
| **(table)** | Displays all the fields for the selected table. Select to add fields to the Where Criteria editor. |
| **=(Equal)** | Select this option to construct an equal (=) comparison between fields. This generates the comparison in ABL code in the Where Criteria editor. |
| **<> (Not Equal)** | Select this option to construct a not equal (<> ) comparison between fields. This generates the comparison in ABL code in the Where Criteria editor. |
| **< (Less Than)** | Select this option to construct a less than (<) comparison between fields. This generates the comparison in ABL code in the Where Criteria editor. |
| **>(Greater Than)** | Select this option to construct a greater than (>) comparison between fields. This generates the comparison in ABL code in the Where Criteria editor. |
| **<= (Less Than or Equal)** | Select this option to construct a less than or equal (<=) comparison between fields. This generates the comparison in ABL code in the Where Criteria editor. |
| **>= (Greater Than or Equal)** | Select this option to construct a greater than or equal (>=) comparison between fields. This generates the comparison in ABL code in the Where Criteria editor. |
| **AND** | Select this option to construct an AND conjunction between two logical expressions in ABL code in the Where Criteria editor. |
| **OR** | Select this option to construct an OR conjunction between two logical expressions in ABL code in the Where Criteria editor. |
| **Begins** | Select this option to create a BEGINS construction for character fields in the Where Criteria editor. |
| **Contains** | Select this option to create a CONTAINS construction for word index fields in the Where Criteria editor. |
| **Matches** | Select this option to create a MATCHES construction for character fields in the Where Criteria editor. |
| **List** | Select this option to create a LIST construction for fields in the Where Criteria editor. |
| **Range** | Select this option to create a RANGE construction for fields in the Where Criteria editor. |
| **Where Criteria** | Displays the where criteria you are creating. You may also edit the query directly in this editor when it has focus. |
Choosetoundothelasteditmade totheWhereCriteria
editor.

Check Syntax Now
Selectthisoptiontohavethecompilercheckthesyntaxof
thequeryinitscurrentstate.

Check Syntax On OK
Selectthisoptiontohavethecompilercheckthesyntaxof
thequerywhenyouclickOK.

Freeform Query
Displays the PROGRESS Advisor dialog which allows
freeform editing of the query using the GUI Designer editor.

See also
QueryBuilderDialogBox -- Table Mode on page 764
QueryBuilderDialogBox -- Join Mode on page 760
QueryBuilderDialogBox -- Sort Mode on page 763
QueryBuilderDialogBox -- Options Mode on page 762

OtherAppBuilderdialogs

Add Directory to Path dialog
The Add Directory to Path dialog allows you to add a pathname that can be accessed from the Choose From
File dialog.

Access this dialog by clicking Add on the Edit Path dialog on page 772.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directory</td>
<td>Enter a valid pathname on your system from which you can access files using the Choose From File dialog on page 768.</td>
</tr>
<tr>
<td>OK</td>
<td>Add the specified pathname to list of accessible paths and return to the Edit Path dialog.</td>
</tr>
</tbody>
</table>

For more information, see Edit Path dialog on page 772

Calculated Field Editor dialog
The Calculated Field Editor dialog allows you to construct a calculated field expression to add to your database browse.

You can access this dialog by clicking Calculated Field or Edit in the Column Editor dialog (for Browse) on page 769.

The Calculated Field Editor dialog includes the following options:
### Expression
Displays the calculated fields expression you build using the options on this dialog box.

**Note:** You can also edit any expression in this editor. To see help for a particular language element, highlight the ABL keyword and press Help.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check Syntax</td>
<td>Select to delete the path highlighted in the list.</td>
</tr>
<tr>
<td>Insert</td>
<td>Select to insert the options selected from Fields, Functions, or Operators into the Expression editor.</td>
</tr>
<tr>
<td>Tables</td>
<td>Select a database table from the drop-down.</td>
</tr>
<tr>
<td>Fields</td>
<td>Select a field to insert into the <strong>Expression</strong> field.</td>
</tr>
<tr>
<td>Functions</td>
<td>Select a function (double-clicking or pressing ENTER) to insert into the <strong>Expression</strong> field.</td>
</tr>
<tr>
<td>Paste Function Arguments</td>
<td>Select this check box to insert argument of the function highlighted in the Functions list. For example, &quot;num&quot; in ABSOLUTE.</td>
</tr>
<tr>
<td>Operators</td>
<td>Select an operator to insert into the <strong>Expression</strong> field.</td>
</tr>
</tbody>
</table>

For more information, see Column Editor dialog (for Browse) on page 769

### Choose From File dialog
The **Choose From File** dialog allows you to locate and insert a particular filename or image.

Access this dialog by selected **Insert from File** from the right-click context menu on the design window.

You can also access the dialog from the main menu by clicking on **Edit > Insert from File**.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Specify a file name or select from the file list.</td>
</tr>
<tr>
<td>File List</td>
<td>List displays the files from the directory specified in Directory.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>If you do not specify a file name in the File field, you can select a file from this list.</td>
</tr>
<tr>
<td>File Type</td>
<td>Select the type of files to display in the file list.</td>
</tr>
<tr>
<td>Directory</td>
<td>Specify the directory to search for files.</td>
</tr>
<tr>
<td>Source (read-only editor)</td>
<td>If the <strong>Preview</strong> check-box is selected, displays the contents of the selected file.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Preview</td>
<td>Select this check-box to display contents of the selected file in the Source field.</td>
</tr>
<tr>
<td>Code/Image</td>
<td>Indicate the type of file being previewed.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The Code and Image options are available only when the Preview check-box is selected.</td>
</tr>
<tr>
<td>Edit Path</td>
<td>Open the Edit Path dialog on page 772 to add, modify, or reorder the paths listed in the Directory field.</td>
</tr>
<tr>
<td>Browse</td>
<td>Open the Browse Files dialog to search for directories and files.</td>
</tr>
</tbody>
</table>

For more information, see GUI Designer context menu on page 693

**Code Sections Selector dialog**

The Code Sections Selector dialog allows you to select a function, internal procedure, or trigger to copy.

Access this dialog by clicking Code on the Copy To File dialog.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available</td>
<td>List the functions, internal procedures, and triggers associated with the selected basic objects and SmartObject instances. Select one or more entries to copy. The selected functions, internal procedures, and triggers move to the Selected list.</td>
</tr>
<tr>
<td>Selected</td>
<td>List the functions, internal procedures, and triggers that you have selected from the Available list to copy. <strong>Note:</strong> You can select one or more fields to remove by clicking &lt;&lt;Remove.</td>
</tr>
<tr>
<td>Add &gt;&gt;</td>
<td>Move the selected functions, internal procedures, and triggers from Available list to the Selected list.</td>
</tr>
<tr>
<td>&lt;&lt;Remove</td>
<td>Move the selected functions, internal procedures, and triggers from the Selected list to Available list.</td>
</tr>
<tr>
<td>OK</td>
<td>Save your changes and return to Copy To File dialog.</td>
</tr>
</tbody>
</table>

For more information, see Copy To File dialog on page 771

**Column Editor dialog (for Browse)**

The Column Editor dialog allows you to add and remove columns associated with database fields, to the selected browse widget.
Access this dialog by clicking **Fields** on the **QueryBuilder** dialog.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fields in Browse</td>
<td>Displays the fields currently defined for the selected browse.</td>
</tr>
<tr>
<td>Add</td>
<td>Select to add fields to the browse definition (accesses the <strong>Multi-Field Selector</strong> dialog box.)</td>
</tr>
<tr>
<td>Remove</td>
<td>Select to remove the highlighted fields in the <strong>Fields in Browse selection</strong> list from the browse definition.</td>
</tr>
<tr>
<td>Move Up</td>
<td>Select to move the highlighted fields in the <strong>Fields in Browse selection</strong> list one line position higher in the list.</td>
</tr>
<tr>
<td>Move Down</td>
<td>Select to move the highlighted fields in the <strong>Fields in Browse selection</strong> list one line position lower in the list.</td>
</tr>
<tr>
<td>Disable-autozap</td>
<td>Select to enable the user to type in the field without the browse automatically clearing the initial value. When unselected, the initial value is erased as soon as the user begins typing in the field.</td>
</tr>
<tr>
<td>Column-Read-Only</td>
<td>Select to enable the user to tab to the column without enabling it for editing.</td>
</tr>
<tr>
<td>Auto-Return</td>
<td>Select to have the browse automatically act as if the user had pressed RETURN after typing the last allowable character in the field.</td>
</tr>
<tr>
<td>Enable</td>
<td>Activate to enable the highlighted fields in the <strong>Fields in Browse selection</strong> list.</td>
</tr>
<tr>
<td>Visible</td>
<td>Select to make the column visible.</td>
</tr>
<tr>
<td>Auto-Resize</td>
<td>Select to enable the browse to automatically resize the column when its FORMAT attribute changes at runtime.</td>
</tr>
<tr>
<td>Enable All</td>
<td>Enables all the fields in the <strong>Fields in Browse selection</strong> list.</td>
</tr>
<tr>
<td>Disable All</td>
<td>Disables all the fields in the <strong>Fields in Browse selection</strong> list.</td>
</tr>
<tr>
<td>Calculated Field</td>
<td>Displays the <strong>Calculated Field Editor</strong> dialog that allows you to add a calculated field to the browse definition.</td>
</tr>
<tr>
<td>Edit</td>
<td>Displays the <strong>Calculated Field Editor</strong> dialog that allows you to add a calculated field to the browse definition.</td>
</tr>
<tr>
<td>Color</td>
<td>Select to change the foreground and background colors for the browse column associated with the highlighted field in the <strong>Fields in Browse selection</strong> list.</td>
</tr>
<tr>
<td>Font</td>
<td>Select to change the font for the browse column associated with the highlighted field in the <strong>Fields in Browse selection</strong> list.</td>
</tr>
<tr>
<td>Translation Attributes</td>
<td>Select to enter translations for the column label and help.</td>
</tr>
<tr>
<td>Feature</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>View-as</strong></td>
<td>Select to change the visualization of the column to a combo-box or toggle-box.</td>
</tr>
<tr>
<td><strong>Label</strong></td>
<td>Enter a label for the browse column associated with the highlighted field in the <strong>Fields in Browse selection</strong> list.</td>
</tr>
<tr>
<td><strong>Label Color</strong></td>
<td>Select to change the foreground and background colors for the browse column label associated with the highlighted field in the <strong>Fields in Browse selection</strong> list.</td>
</tr>
<tr>
<td><strong>Label Font</strong></td>
<td>Select to change the font for the browse column label associated with the highlighted field in the <strong>Fields in Browse selection</strong> list.</td>
</tr>
<tr>
<td><strong>Format</strong></td>
<td>Enter the display format for the browse column associated with the highlighted field in the <strong>Fields in Browse selection</strong> list.</td>
</tr>
<tr>
<td><strong>Format Help</strong></td>
<td>Select to access the <strong>Format</strong> dialog box for assistance in setting the column display format.</td>
</tr>
<tr>
<td><strong>Width</strong></td>
<td>Enter the maximum characters for the column.</td>
</tr>
<tr>
<td><strong>Column Help</strong></td>
<td>Enter a help string for the browse column associated with the highlighted field in the <strong>Fields in Browse selection</strong> list. At run time, this help string will be displayed in the window status area whenever the browse column is selected.</td>
</tr>
</tbody>
</table>

**See also**
- Choose Color dialog on page 753
- Choose Font dialog on page 754
- String Attributes dialog on page 782
- View-as attributes dialog on page 784
- Calculated Field Editor dialog on page 767

**Copy To File dialog**

The **Copy To File** dialog allows you to copy basic objects and SmartObject instances and their associated internal procedures, functions, and triggers to an export (.wx) file.

Access this dialog by selecting **Copy to File** from the right-click context menu on the design window.

You can also access the dialog from the main menu by clicking on **Edit > Copy to File**.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Items to Copy</td>
<td>List basic objects and SmartObject instances selected in the current design window.</td>
</tr>
<tr>
<td>Code</td>
<td>Open the <strong>Code Sections Selector dialog</strong> on page 769 which allows you to select internal procedures associated with the selected objects to copy.</td>
</tr>
<tr>
<td>OK</td>
<td>Save your changes and return to the design window.</td>
</tr>
</tbody>
</table>
For more information, see GUI Designer context menu on page 693

**Data Dictionary window**

The Data Dictionary is a tool that lets you create and modify database schema information including table, field, sequence, index, and trigger definitions.

**Note:** If you started the Data Dictionary without connecting to a database, only the Databases icons, an empty Database selection list, and the Create Database button are displayed. has functionality that is similar to DB Navigator. Consider using DB Navigator since it is integrated with Progress Developer Studio and is therefore more accessible to your project.

To start the Data Dictionary, select **OpenEdge > Admin > Data Dictionary** from the menu of any OpenEdge perspective except DB Navigator and Debug.

When started, the Data Dictionary runs in a window outside the Progress Developer Studio Eclipse framework. Online help is available from the tool's menu bar.

**Edit Path dialog**

The **Edit Path** dialog allows you to add, remove, or reorder the paths listed in the **Directory** field on the **Choose From File** dialog.

Access this dialog by clicking **Edit Path** on the **Choose From File** dialog.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>List</td>
<td>Lists the paths displayed in the <strong>Directory</strong> field on the <strong>Choose From File</strong> dialog.</td>
</tr>
<tr>
<td>Add</td>
<td>Open the <strong>Add Directory to Path</strong> dialog on page 767 which allows you to add new paths to the list.</td>
</tr>
<tr>
<td>Remove</td>
<td>Delete the selected path from the list.</td>
</tr>
<tr>
<td>Move Up</td>
<td>Choose to move the selected path one line position up in the list.</td>
</tr>
<tr>
<td>Move Down</td>
<td>Choose to move the selected path one line position down in the list.</td>
</tr>
<tr>
<td>OK</td>
<td>Save your changes and return to <strong>Choose From File</strong> dialog.</td>
</tr>
</tbody>
</table>

For more information, see **Choose From File** dialog on page 768

**Enter Constant** dialog

The **Enter Constant** dialog allows you to input a value constant for an operation.

This dialog includes the following option:

| Enter data-type Value | Enter a valid value. You must enter a value of the data type for the field to accept the value.                                      |

For more information, see Query Builder dialog
Enter Constant Range dialog

The *Enter Constant Range* dialog allows you to input the range for an operation.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter Lower Bound</td>
<td>Enter a smaller value for the range. You must use the proper data type for the field to accept the value.</td>
</tr>
<tr>
<td>Enter Upper Bound</td>
<td>Enter a larger value for the range. You must enter a value of the data type requested for the field to accept the value.</td>
</tr>
<tr>
<td>Inclusive</td>
<td>Select this check box to include the boundary conditions.</td>
</tr>
</tbody>
</table>

For more information, see *QueryBuilder dialog*

Enter Contain String dialog

The *Enter Contain String* dialog allows you to query fields that have word indexes.

This dialog includes the following option:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| Enter String Value   | Enter a valid search-expression with which to construct a CONTAINS phrase for the selection criteria in the Where Builder dialog box. The search-expression specifies one or more words to search for. It must evaluate to a string with the following syntax:

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

In the syntax above, all metasyntax characters are in bold. Do not type metasyntax characters when entering a search expression. Each word is a word to search for. The ampersand (&) represents a logical AND; the vertical line (|), exclamation point (!), or caret (^) represent a logical OR. |

**Note:** See also the WHERE clause of the Record phrase.

See also

*QueryBuilder dialog* on page 760

Enter List of Constants dialog

The *Enter List of Constants* dialog allows you to input a list of values for an operation.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter data-type value</td>
<td>Enter a value. You must use the proper data type for the field to accept the value.</td>
</tr>
<tr>
<td>Add</td>
<td>Append the value entered in the <em>Enter (data-type) Value</em> field to the list of values for the operation.</td>
</tr>
<tr>
<td>Change</td>
<td>Modify a entry highlighted in the selection list with the value entered in the <em>Enter (data-type) Value</em> field.</td>
</tr>
</tbody>
</table>
Remove | Delete an entry highlighted in the selection list above.
List | Displays the entered values. Highlight a value to change or to remove.
Inclusive | Select this check box to display only the records that equal the criteria displayed in the list.

For more information, see Query Builder dialog

**Folder Selection dialog**

The **Folder Selection** dialog allows you to select an OpenEdge project to which you want to add the new ABL GUI procedure (.w) file.

Access this dialog by clicking **Browse** on the **Create an ABL GUI procedure** wizard.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Select new file container</th>
<th>Displays the list of projects present in your current workspace.</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td>Saves the selected project and returns you to the <strong>Create an ABL GUI procedure</strong> wizard.</td>
</tr>
</tbody>
</table>

For more information, see **Creating ABL GUI procedure files** on page 644

**Format dialog for Character field**

The **Format** dialog helps you to format the selected **Character** field.

You can access this dialog by clicking **Format Help** in the **Column Editor dialog (for Browse)** on page 769.

The **Format** dialog for **Character** field includes the following options:

<table>
<thead>
<tr>
<th>Character Format</th>
<th>Displays a valid display format for the selected field.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Width</td>
<td>Displays the character width for the selected field. You can change the value by entering a different width value.</td>
</tr>
<tr>
<td>Allow Any Character</td>
<td>Select to allow any character in the string format. (including special characters.)</td>
</tr>
<tr>
<td>Allow Letters and Numbers Only</td>
<td>Select to allow only numbers and letters in the string format.</td>
</tr>
<tr>
<td>Allow Letters Only</td>
<td>Select to allow only letters in the string format.</td>
</tr>
<tr>
<td>Allow Letters and Convert to Caps</td>
<td>Select to allow only letters and display them in uppercase in the string format.</td>
</tr>
<tr>
<td>Allow Numbers Only</td>
<td>Select to allow only numbers in the string format.</td>
</tr>
<tr>
<td>Standard</td>
<td>Select to format the selected field to the default ABL format for strings, X(8).</td>
</tr>
</tbody>
</table>

**See also**

Format dialog for Logical field on page 778
Formatting a Date field

The **Format** dialog helps you to format the selected Date field. You can access this dialog by clicking **Format Help** in the **Column Editor dialog (for Browse)** on page 769. The **Format** dialog for **Date** field includes the following options:

<table>
<thead>
<tr>
<th>Date Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>99/99/99</td>
<td>Select to display the date in two-byte segments delimited by a backslash (/). This is the standard format.</td>
</tr>
<tr>
<td>99-99-99</td>
<td>Select to display the date in two-byte segments delimited by a hyphen (-).</td>
</tr>
<tr>
<td>99.99.99</td>
<td>Select to display the date in two-byte segments delimited by a period (.).</td>
</tr>
<tr>
<td>99/99/9999</td>
<td>Select to display the date in two-byte segments for the month and day and display the year in four-bytes. Delimit with a backslash (/).</td>
</tr>
<tr>
<td>99-99-9999</td>
<td>Select to display the date in two-byte segments for the month and day and display the year in four-bytes. Delimit with a hyphen (-).</td>
</tr>
<tr>
<td>99.99.9999</td>
<td>Select to display the date in two-byte segments for the month and day and display the year in four-bytes. Delimit with a period (.).</td>
</tr>
<tr>
<td>999999</td>
<td>Select to display the date in six bytes with no delimiter character.</td>
</tr>
<tr>
<td>99999999</td>
<td>Select to display the date in eight bytes with no delimiter character.</td>
</tr>
<tr>
<td>Standard</td>
<td>Select to format the selected field to the default ABL format for dates, 99/99/99.</td>
</tr>
</tbody>
</table>

**See also**

- Format dialog for Character field on page 774
- Format dialog for Numeric field on page 778
- Format dialog for Date Time field on page 777
- Format dialog for Date Time-Tz field on page 775
- Format dialog for Logical field on page 778

Formatting a Date Time-Tz field

The **Format** dialog helps you to format the selected **Date Time-Tz** field. You can access this dialog by clicking **Format Help** in the **Column Editor dialog (for Browse)** on page 769. The **Format** dialog for **Date Time-Tz** field includes the following options:

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Enter a valid display format for the selected <strong>Date Time-Tz</strong> field.</td>
</tr>
<tr>
<td>Format</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>99/99/99</td>
<td>Select to display the date portion of the DateTime-Tz field in two-byte segments delimited by a backslash (/). This is the standard format.</td>
</tr>
<tr>
<td>99-99-99</td>
<td>Select to display the date portion of the DateTime-Tz field in two-byte segments delimited by a hyphen (-).</td>
</tr>
<tr>
<td>99.99.99</td>
<td>Select to display the date portion of the DateTime-Tz field in two-byte segments delimited by a period (.).</td>
</tr>
<tr>
<td>99/99/9999</td>
<td>Select to display the date portion of the DateTime-Tz field in two-byte segments for the month and day and display the year in four-bytes. Delimit with a backslash (/).</td>
</tr>
<tr>
<td>99-99-9999</td>
<td>Select to display the date portion of the DateTime-Tz field in two-byte segments for the month and day and display the year in four-bytes. Delimit with a hyphen (-).</td>
</tr>
<tr>
<td>99.99.9999</td>
<td>Select to display the date portion of the DateTime-Tz field in two-byte segments for the month and day and display the year in four-bytes. Delimit with a period (.).</td>
</tr>
<tr>
<td>999999</td>
<td>Select to display the date portion of the DateTime-Tz field in six bytes with no delimiter character.</td>
</tr>
<tr>
<td>99999999</td>
<td>Select to display the date portion of the DateTime-Tz field in eight bytes with no delimiter character.</td>
</tr>
<tr>
<td>Include Time</td>
<td>Select this option when you want to include the time portion in the format.</td>
</tr>
</tbody>
</table>

**Time Format**

Select this option when you want to include a time zone in the format. Include TimeZone

**Decimal Places for Seconds**

Select the number of decimal places you want for the second’s portion of the time format. You can select 0, 1, 2, or 3.

**12 hour format (AM/PM)**

Select to specify a 12 hour time format. When you select this format hours are displayed using AM or PM.

**Include Time Zone**

Select when you want to include a time zone in the format.

**Standard**

Select to format the selected field to the default ABL Date Time-Tz format: 99/99/9999 HH:MM:SS.SSS+HH:MM.

**Note:** These options are only available when you select Include time.
See also
Format dialog for Character field on page 774
Format dialog for Numeric field on page 778
Format dialog for Date Time field on page 777
Format dialog for Logical field on page 778
Format dialog for Date field on page 775

Formatting a Date Time field

The **Format** dialog helps you to format the selected **Date Time-Tz** field.

You can access this dialog by clicking **Format Help** in the Column Editor dialog (for Browse) on page 769.

The **Format** dialog for **Date Time-Tz** field includes the following options:

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>99/99/99</td>
<td>Select to display the date portion in two-byte segments delimited by a backslash (/). This is the standard format.</td>
</tr>
<tr>
<td>99-99-99</td>
<td>Select to display the date portion in two-byte segments delimited by a hyphen (-).</td>
</tr>
<tr>
<td>99.99.99</td>
<td>Select to display the date portion in two-byte segments delimited by a period (.).</td>
</tr>
<tr>
<td>99/99/9999</td>
<td>Select to display the date portion in two-byte segments for the month and day and display the year in four-bytes. Delimit with a backslash (/).</td>
</tr>
<tr>
<td>99-99-9999</td>
<td>Select to display the date portion in two-byte segments for the month and day and display the year in four-bytes. Delimit with a hyphen (-).</td>
</tr>
<tr>
<td>99.99.9999</td>
<td>Select to display the date portion in two-byte segments for the month and day and display the year in four-bytes. Delimit with a period (.).</td>
</tr>
<tr>
<td>99999999</td>
<td>Select to display the date portion in six bytes with no delimiter character.</td>
</tr>
<tr>
<td>9999999999</td>
<td>Select to display the date portion in eight bytes with no delimiter character.</td>
</tr>
<tr>
<td><strong>Include Time</strong></td>
<td>Select when you want to include the time portion in the format.</td>
</tr>
</tbody>
</table>

**Time Format**

Specify a time format. The option you select determines how the time portion of the format appears. You can select:

- **HH** - Displays only the hour.
- **HH:MM** - Displays the hour and minutes.
- **HH:MM:SS** - Displays the hours, minutes, and seconds.

**Note:** These options are only available when you select **Include time**.
Decimal Places for Seconds | Select the number of decimal places you want for the second’s portion of the time format. You can select 0, 1, 2, or 3.

| **Note:** These options are only available when you select **Include time**.

12 hour format (AM/PM) | Select to specify a 12 hour time format. When you select this format hours are displayed using AM or PM.

Include Time Zone | Select when you want to include a time zone in the format.

Standard | Select to format the selected field to the default ABL DateTime format: 99/99/9999 HH:MM:SS.SSS+HH:MM.

See also
Format dialog for Character field on page 774
Format dialog for Numeric field on page 778
Format dialog for Logical field on page 778
Format dialog for Date Time-Tz field on page 775
Format dialog for Date field on page 775

Format dialog for Logical field
The Format dialog helps you to format the selected Logical field.
You can access this dialog by clicking Format Help in the Column Editor dialog (for Browse) on page 769.
The Format dialog for Logical field includes the following options:

| Logical Format | Enter a valid display format for the selected logical field (True or False).

| Display This When TRUE | Enter a string to display when the field contains a TRUE value such as Yes.

| Display This When FALSE | Enter a string to display when the field contains a FALSE value such as No.

| Standard | Select to format the selected field to the default ABL format for logicals, yes/no.

See also
Format dialog for Character field on page 774
Format dialog for Numeric field on page 778
Format dialog for Date Time field on page 777
Format dialog for Date Time-Tz field on page 775
Format dialog for Date field on page 775

Format dialog for Numeric field
The Format dialog helps you to format the selected Numeric field.
You can access this dialog by clicking Format Help in the Column Editor dialog (for Browse) on page 769.
The **Format** dialog for **Numeric** field includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number Format</td>
<td>Enter a valid display format for the selected numeric field (-&gt;,&gt;&gt;,&gt;&gt;9 is the default).</td>
</tr>
<tr>
<td>Total Number of Digits</td>
<td>Enter the total number of digits allowed for the selected field (7 is the default).</td>
</tr>
<tr>
<td>Number of Decimal Places</td>
<td>Enter the number of decimal places to the right of the decimal character to display for the field value.</td>
</tr>
<tr>
<td>Use Thousand's Separator</td>
<td>Select to display a comma (,) as the thousands separator character for the field value (default).</td>
</tr>
<tr>
<td>Suppress Leading Zeros</td>
<td>Select to suppress any leading zeros in the field value.</td>
</tr>
<tr>
<td>Show as Zero</td>
<td>Select to display zero values in the field as blank.</td>
</tr>
<tr>
<td>Show as Asterisk</td>
<td>Select to display an asterisk (*) as the leading sign for the field value.</td>
</tr>
<tr>
<td>Leading Text String</td>
<td>Enter a text string to display before the number, such as a currency sign.</td>
</tr>
<tr>
<td>Trailing Text String</td>
<td>Enter a text string to display after the number, such as a currency sign.</td>
</tr>
<tr>
<td>Show Leading Sign Always</td>
<td>Select to display all values with a leading sign.</td>
</tr>
<tr>
<td>Show Leading Sign on Negative Only</td>
<td>Select to display only negative values with the leading sign (default).</td>
</tr>
<tr>
<td>Show Trailing Sign Always</td>
<td>Select to display all values with a trailing sign.</td>
</tr>
<tr>
<td>Show Trailing Sign on Negative Only</td>
<td>Select to display only negative values with the trailing sign.</td>
</tr>
<tr>
<td>Show Negative Numbers in Parentheses</td>
<td>Select to display negative values between parentheses '()'.</td>
</tr>
<tr>
<td>Show Negatives With Trailing 'CR'</td>
<td>Select to display negative values with the string &quot;CR&quot; after it.</td>
</tr>
<tr>
<td>Show Negatives With Trailing 'DB'</td>
<td>Select to display negative values with the string &quot;DB&quot; after it.</td>
</tr>
<tr>
<td>Show Negatives With Trailing 'DR'</td>
<td>Select to display negative values with the string &quot;DR&quot; after it.</td>
</tr>
<tr>
<td>Show Negatives With Trailing 'cr'</td>
<td>Select to display negative values with the string &quot;cr&quot; after it.</td>
</tr>
<tr>
<td>Show Negatives With Trailing 'db'</td>
<td>Select to display negative values with the string &quot;db&quot; after it.</td>
</tr>
<tr>
<td>Show Negatives With Trailing 'dr'</td>
<td>Select to display negative values with the string &quot;dr&quot; after it.</td>
</tr>
</tbody>
</table>
Allow Positive Numbers Only
- Select to allow positive numbers only.

Standard
- Select to format the selected field to the default ABL format for numbers, which is: ->, >>>, >>9, 7 total digits, comma as thousands separator, and show leading sign on negatives only.

See also
- Format dialog for Character field on page 774
- Format dialog for Date field on page 775
- Format dialog for Date Time field on page 777
- Format dialog for Date Time-Tz field on page 775
- Format dialog for Date field on page 775

Goto Page dialog
The Goto Page dialog allows you to display change the page displayed in the design window.
A page is a logical grouping of SmartObjects which are manipulated in unison. When you view a given page, the ADM automatically hides all other pages.

Note: The Main page, page 0, is always displayed along with the selected page.

Pages provide an easy to use mechanism for viewing and hiding objects, and also conserve screen real estate. All SmartContainers can manage application pages. You can dispatch any ADM method to the entire group. Access this dialog by selecting Goto Page from right-click context menu on the design window. Alternatively, you can use the Page option from the main toolbar.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display Page Number</td>
<td>Enter a page number to display in the design window.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The &quot;Main&quot; page (page 0) is always displayed in the design window, in addition to the page selected here.</td>
</tr>
<tr>
<td>OK</td>
<td>Add the selected page to the SmartObject instance on the design window.</td>
</tr>
</tbody>
</table>

See also
- Paging in SmartObjects on page 638
- GUI Designer context menu on page 693

Method Libraries dialog
The Method Libraries dialog allows you to add, remove, or modify references to method libraries of the procedure.
You can access this dialog by clicking the **Method Libraries** icon on the **Procedure Settings** dialog. This dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method Library Include References</td>
<td>Displays the method libraries included with the procedure.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The nested method libraries are not displayed in the selection list.</td>
</tr>
<tr>
<td>Add</td>
<td>Displays the <strong>Add Method Library Reference</strong> dialog which allows you to add a reference to a method library.</td>
</tr>
<tr>
<td>Modify</td>
<td>Displays the <strong>Modify Method Library Reference</strong> dialog which allows you to change a reference to a method library.</td>
</tr>
<tr>
<td>Remove</td>
<td>Deletes a reference to a method library.</td>
</tr>
<tr>
<td>Move Up</td>
<td>Moves the selected method library reference one position up in the list.</td>
</tr>
<tr>
<td>Move Down</td>
<td>Moves the selected method library reference one position down in the list.</td>
</tr>
</tbody>
</table>

**See also**

*About Method Library on page 639*
*Adding method library reference on page 671*
*Modifying method library reference on page 671*

**Record Menu Accelerators dialog**

The **Record Menu Accelerators** dialog allows you to record accelerator keys for menu commands. Accelerator key is a key or key combination which when pressed invokes an associated command.

When the dialog is first opened, you are in record mode. The keystrokes you enter are recorded as the accelerator key for the displayed menu command. You can use **Stop Record/Record** alternately to stop and to start recording an accelerator key.

Access this dialog by clicking **Key** on the **Popup Menu or Menu Bar** dialog.

The **Record Menu Accelerators** dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accelerator Key</td>
<td>Enter a key or key combination. Displays the accelerator keys you just defined for the menu command displayed in the Menu Item field.</td>
</tr>
<tr>
<td>Cancel</td>
<td>Discard any menu accelerator keys recorded and close the dialog box. You can also choose Cancel to stop recording and to close the dialog box.</td>
</tr>
<tr>
<td>Menu Item</td>
<td>Displays the menu command for which you are recording an accelerator key.</td>
</tr>
<tr>
<td>Stop Record/Record</td>
<td>Select alternately to stop and to start recording an accelerator key.</td>
</tr>
</tbody>
</table>

For more information, see *Property Sheet for Menu Bar or Popup Menu* on page 705
**String Attributes dialog**

The **String Attributes** dialog helps you to set translation attributes for the selected object.

**Note:** The controls displayed on this dialog is based on the selected object.

You can access this dialog by clicking on the **Translation Attributes** icon on the object's property sheet dialog or Column Editor dialog (for Browse) on page 769.

The **String Attributes** dialog includes the following options:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Displays the title of the selected object.</td>
</tr>
<tr>
<td>Tooltip</td>
<td>Displays the tooltip assigned to the selected object.</td>
</tr>
<tr>
<td>Label</td>
<td>Displays the label for the selected object.</td>
</tr>
<tr>
<td>Format</td>
<td>Displays the display format for the selected object.</td>
</tr>
<tr>
<td>Help</td>
<td>Displays the help string for the selected object.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Displays valid character-string attributes for each of the object's characteristics such as Title, Tooltip, Label, Format, or Help.</td>
</tr>
</tbody>
</table>

**OK**

Select to save your changes and exit this dialog.

**Cancel**

Select to ignore your changes and exit this dialog.

For more information, see **Selecting translation attributes for an object** on page 654

**Sync with Master dialog**

The **Sync with Master** dialog allows you to re-establish connections between the master and alternate layouts.

Access this dialog by clicking **Sync with Master** on the selected object's **Advanced Properties** dialog.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>List</td>
<td>A tabular list of all the attribute discrepancies between the master layout and the alternate layout for the selected object.</td>
</tr>
<tr>
<td>Master Updates to Alternate Value</td>
<td>Select this option to have the master layout adopt the attributes of the alternate layout.</td>
</tr>
<tr>
<td>Alternate Reverts to Master Value</td>
<td>Select this option to have the alternate layout adopt the attributes of the master layout.</td>
</tr>
</tbody>
</table>

For more information, see **Creating an Alternate Layout** on page 651
**Temp-Table Maintenance Dialog Box**

Use this dialog box to add, remove, and modify Temp-table definitions within the current procedure file. You can select tables in your TEMP-DB or tables in any existing database. You can also modify existing table schema by adding optional fields and indexes. (Access this dialog box by choosing the Temp-Tables icon in the Procedure Settings dialog.)

**Note:** All items on the Temp-tables Maintenance box are grayed out until you select a table using the Add button.

| Add | Displays the **Table Selector** dialog. From this box select any tables not yet selected for the current procedure. After you select a table, the AppBuilder displays it in the Temp-Tables Maintenance box and activates the other options.  

**Note:** The AppBuilder uses the ABL "LIKE" syntax to define the Temp-Table. Defining the Temp-Table like an existing table causes the Temp-Table definition to be stored in the code during compilation, but does not require the TEMP-DB to be connected at run time. This is necessary because Temporary Tables do not exist until run-time. |
|-----|---|
| Remove | Removes a table definition from the current procedure after you highlight it in the display box.  
To apply table properties or other options to a table, highlight it from the displayed list, then select items as described below. |
| New Global Shared | Establishes a global shared Temporary Table. The scope of a global shared Temporary Table is the session. |
| New Shared | Establishes a shared Temporary Table. The scope of a shared Temporary Table is the scope of the procedure that established it. |
| Shared | Allows the procedure to access an established shared Temporary Table. |
| Local | Allows access only from this procedure. This is the default selection. |
| Buffer | Defines an extra buffer for the table instead of defining a Temporary Table. As with the Temporary Table, the scope of the buffer is determined by your selection from the properties listed above. For example, if you select "New Shared" and "Buffer", you are defining a new shared buffer. |
| NO-UNDO | Increases update performance for Temporary Table. If you deselect this button, changes made to the Temporary Table during a transaction are logged to the local before image (LBI) file. |
| Table Name | Allows you to rename your Temporary Table. The new name is updated in the Temp-Tables list as soon as you leave the fill-in. |
**Additional Fields**

Allows you to add extra fields to a defined Temp-Table. For example, you might want to maintain the schema of an existing database table, but add one or two extra fields. You can also use this fill-in to override the primary index for the table.

**Note:** Fields entered in Additional fields will not be available for selection in the Query Builder and other AppBuilder schema pickers. Additional Fields is reserved for less visible data, such as ROWID or a time stamp. Fields that you want available from the schema pickers should be defined in the Data Dictionary.

**Check Syntax**

Checks the syntax of any entries in Additional Fields prior to running or saving your procedure.

---

**View-as attributes dialog**

The **View-as attributes** dialog allows you to change the default visualization of a column in a browser widget.

**Note:** This dialog behaves differently depending on whether you are working with a static or dynamic SmartDataBrowser.

You can access this dialog by clicking on the **View-as** icon on the object's property sheet dialog or Column Editor dialog (for Browse) on page 769.

The **View-as attributes** dialog includes the following options:

<table>
<thead>
<tr>
<th>Field</th>
<th>Displays the field selected in the <strong>Field in Browse</strong> list on the <strong>Column Editor</strong> dialog. This drop-down is a selection list of the columns in the browse widget.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>The <strong>Field</strong> drop-down is enabled only with SmartDataBrowser that is linked to a data object.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>View-as</th>
<th>Specify how the selected field must appear in the browse widget. The options available are: <strong>Fill-in</strong> and <strong>Combo-box</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>Fill-in is the default value.</td>
</tr>
</tbody>
</table>

| List-Item   | Select one of the following option:  
|-------------|--------------------------------------------------------------------------------------|
| **• List-Items** - to display single items in the combo-box such as Item1 and Item2.  
**• List-Item-Pairs** - to display items in pairs in the combo-box such as Item1, Item2 and Item3, Item4. |
| **Note:**   | These options are available when you select combo-box from the **View-as** drop-down. |

---
<table>
<thead>
<tr>
<th><strong>List-Item Editor</strong></th>
<th>Enter the items to display in the combo-box.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong> This option is available when you select combo-box from the <strong>View-as</strong> drop-down.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Type</strong></th>
<th>Select either <strong>DROP-DOWN-LIST</strong> or <strong>DROP-DOWN</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong> <strong>DROP-DOWN-LIST</strong> is the default value.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Printable</strong></th>
<th>This is delimiter type. When you select this option, you must enter a printable key such as a comma.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Non-printable</strong></th>
<th>This is delimiter type. When you select this option, you must enter the CHR number for that character such as 1 for CHR(1).</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Delimiter</strong></th>
<th>The character that separates values input to or output from the combo-box. The valid values for this field depend on the delimiter type setting: Printable or Non-printable.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Inner Lines</strong></th>
<th>Enter a value that specifies the number of items visible at one time in the list.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong> This option is available when you select combo-box from the <strong>View-as</strong> drop-down.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Maximum Characters</strong></th>
<th>The maximum number of characters for an item or item-pair.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong> This option is available when you select <strong>DROP-DOWN</strong> from the <strong>Type</strong> list.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Sort</strong></th>
<th>Select this check-box to insert all items in the combo-box in the sorted order.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong> This option is available when you select <strong>Combo-box</strong> in the <strong>View-as</strong> drop-down.</td>
<td></td>
</tr>
</tbody>
</table>
Auto-Completion
Select this check-box to automatically complete keyboard input to the first match to items in the combo-box.

Note: This option is available when you select DROP-DOWN from the Type list.

Unique-Match
Select this check-box to automatically complete the keyboard input when it finds a unique match to items in the combo-box.

Note: This option is available when you select the Auto-Completion check-box.

For more information, see Column Editor dialog (for Browse) on page 769

Progress Advisor (Link Conflict)
You see the Progress Advisor because you are trying to add a SmartLink that is unsupported in the Target object.

Choose from the radio set whether to disregard the Advisor warning and create the SmartLink, or cancel creation of the SmartLink.

See also
SmartLinks dialog on page 805

Custom Lists dialog box
Use the Custom Lists dialog box to change the names of the list preprocessor names. To change a name, overwrite the old name and choose OK.

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>List-1</td>
<td>Enter a name for the List-1 preprocessor list.</td>
</tr>
<tr>
<td>List-2</td>
<td>Enter a name for the List-2 preprocessor list.</td>
</tr>
<tr>
<td>List-3</td>
<td>Enter a name for the List-3 preprocessor list.</td>
</tr>
<tr>
<td>List-4</td>
<td>Enter a name for the List-4 preprocessor list.</td>
</tr>
<tr>
<td>List-5</td>
<td>Enter a name for the List-5 preprocessor list.</td>
</tr>
<tr>
<td>List-6</td>
<td>Enter a name for the List-6 preprocessor list.</td>
</tr>
</tbody>
</table>

See also
Procedure Settings dialog box on page 700
External Tables dialog box

Use the **External Tables** dialog box to add external tables to a SmartObject.

An external table supplies records to a query that the query need to complete its search criteria. For example, a query on Order OF Customer requires the external table Customer.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Editor)</td>
<td>Displays the list of selected tables in the connected database. (The form shown is database-name.table-name.)</td>
</tr>
<tr>
<td>Add</td>
<td>Choose to add an external table to the SmartObject. You go to the <strong>Table Selector</strong> dialog box.</td>
</tr>
<tr>
<td>Delete</td>
<td>Choose to remove the highlighted tables from the SmartObject.</td>
</tr>
<tr>
<td>Move Up</td>
<td>Choose to move the highlighted table one line position higher in the list.</td>
</tr>
<tr>
<td>Move Down</td>
<td>Choose to move the highlighted table one line position lower in the list.</td>
</tr>
</tbody>
</table>

See also

**Table Selector dialog box** on page 756

Record Menu Accelerators dialog box

Use the **Record Menu Accelerators** dialog box to record accelerator keys for menu commands. Accelerator key is a key or key combination which when pressed invokes an associated command.

When this dialog is first opened, you are in record mode. The keystrokes you enter are recorded as the accelerator key for the displayed menu command. You can **Stop Record/Record** alternately to stop and to start recording an accelerator key.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accelerator Key</td>
<td>Enter a key or key combination. Displays the accelerator keys you just defined for the menu command displayed in the <strong>Menu Item</strong> field.</td>
</tr>
<tr>
<td>Cancel</td>
<td>Discard any menu accelerator keys recorded and close the dialog box. You can also choose <strong>Cancel</strong> to stop recording and to close the dialog box.</td>
</tr>
<tr>
<td>Menu Item</td>
<td>Displays the menu command for which you are recording an accelerator key.</td>
</tr>
<tr>
<td>Stop Record/Record</td>
<td>Choose alternately to stop and to start recording an accelerator key.</td>
</tr>
</tbody>
</table>
See also
Property Sheet for Popup Menu dialog box on page 705

SmartObject dialogs

Open SmartObject dialog

The Open SmartObject dialog allows you to select a SmartObject master to use to create and insert a SmartObject instance on the design canvas.

Access the Open SmartObject dialog by selecting a SmartObject from the SmartObjects section in the Palette.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master File</td>
<td>Select a master file from a selection list or enter a name of the SmartObject master file you want to use to create a SmartObject instance.</td>
</tr>
<tr>
<td>Preview</td>
<td>Enable you to view the selected SmartObject master in the Preview SmartObject dialog box on page 808, where the GUI Designer runs the SmartObject.</td>
</tr>
<tr>
<td>List</td>
<td>Display a list of SmartObject master files.</td>
</tr>
<tr>
<td>Browse</td>
<td>Open the browse files dialog that allows you to search and select a SmartObject master file in other directories.</td>
</tr>
<tr>
<td>New</td>
<td>Select to create a new SmartObject master of the type selected.</td>
</tr>
<tr>
<td>File Filter</td>
<td>Display the file filter used to display the filenames shown in the Master File field. You can select another file filter by highlighting one from the drop-down list.</td>
</tr>
<tr>
<td>Directory</td>
<td>Display the pathname containing the SmartObject master files. You can select another directory by highlighting one from the drop-down.</td>
</tr>
</tbody>
</table>

See also
About SmartObjects on page 637
Inserting SmartObjects on page 669
List of SmartObjects on page 684

SmartDataObject Wizard

The SmartDataObject Wizard allows you to create a SmartDataObject (SDO) object. You can use the wizard to define a query to retrieve data from the connected database, and define a set of field values to make available with the SmartDataObject you create.

Access the SmartDataObject wizard by choosing SmartDataObject in the Object Type list on the Create an ABL GUI procedure dialog.

The SmartDataObject Wizard is a five page wizard and includes the following options:
<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SmartDataObject Wizard - Page 1 of 5</strong></td>
<td></td>
</tr>
<tr>
<td>&gt;Next</td>
<td>Move to the next page of the wizard.</td>
</tr>
<tr>
<td><strong>SmartDataObject Wizard - Page 2 of 5</strong></td>
<td></td>
</tr>
<tr>
<td>Use SDO logic procedure</td>
<td>Select to specify the SmartDataObject's logic procedure that will contain the business logic.</td>
</tr>
<tr>
<td>Generate new DLP</td>
<td>Select to generate a new Data Logic Procedure (DLP) and associate with the SmartDataObject.</td>
</tr>
<tr>
<td>Note: In the Logic procedure file name field, specify a path where you want to save the new DLP.</td>
<td></td>
</tr>
<tr>
<td>Attach existing DLP</td>
<td>Select to associate an existing DLP with the SmartDataObject.</td>
</tr>
<tr>
<td>Note: In the Logic procedure file name field, specify the path of an existing DLP.</td>
<td></td>
</tr>
<tr>
<td>Logic procedure file name</td>
<td>Specify a path for the new DLP or a path of an existing DLP. Click <strong>Browse</strong> to select a different path for the new DLP or select for an existing DLP.</td>
</tr>
<tr>
<td>Use template</td>
<td>Specify a relative directory and filename for the new generated DLP.</td>
</tr>
<tr>
<td>Note: The Use template field is enabled only when the Generate new DLP option is selected.</td>
<td></td>
</tr>
<tr>
<td>Use NO-UNDO for RowObject</td>
<td>Select this option when defining RowObject temp table. It must be defined NO-UNDO, if BLOB or CLOB fields are included in the SmartDataObject.</td>
</tr>
<tr>
<td><strong>SmartDataObject Wizard - Page 3 of 5</strong></td>
<td></td>
</tr>
<tr>
<td>Define Temp-Tables</td>
<td>Open the Temp-Table Maintenance dialog which allows you to add, remove, and modify Temp-table definitions within your current procedure file. You can select tables in your TEMP-DB or tables in any existing database. You can also modify existing table schema by adding optional fields and indexes.</td>
</tr>
<tr>
<td>Define Query</td>
<td>Open the Query Builder dialog which allows you to define a query that you want to use in the SmartDataObject.</td>
</tr>
<tr>
<td>Note: If the query involves temp-tables, you must define them first using Define Temp-Tables.</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Help on Queries</td>
<td>Open the help on the AppBuilder queries.</td>
</tr>
<tr>
<td><strong>SmartDataObject Wizard - Page 4 of 5</strong></td>
<td></td>
</tr>
<tr>
<td>Add Fields</td>
<td>Open the Column Editor dialog which allows you to add a list of fields to the SmartDataObject to make available to visualization objects.</td>
</tr>
<tr>
<td>Help on DataObject</td>
<td>Open the help on fields.</td>
</tr>
<tr>
<td><strong>SmartDataObject Wizard - Page 5 of 5</strong></td>
<td></td>
</tr>
<tr>
<td>Finish</td>
<td>Complete the wizard to create a SmartDataObject.</td>
</tr>
</tbody>
</table>

For more information, see Creating ABL GUI procedure files on page 644

**SmartDataBrowser Wizard**

The SmartDataBrowser Wizard allows you to create a SmartDataBrowser (SDB) object. The wizard allows you to first choose a DataSource to supply data to SmartDataBrowser, and then you specify columns that will be displayed in the SmartDataBrowser object.

Access the SmartDataBrowser wizard by choosing SmartDataBrowser in the Object Type list on the Create an ABL GUI procedure dialog.

The SmartDataBrowser Wizard is a four page wizard and includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SmartDataBrowser Wizard - Page 1 of 4</strong></td>
<td></td>
</tr>
<tr>
<td>Next</td>
<td>Move to the next page of the wizard.</td>
</tr>
<tr>
<td><strong>SmartDataBrowser Wizard - Page 2 of 5</strong></td>
<td></td>
</tr>
<tr>
<td>Data definition source</td>
<td>Specify a DataObject (SmartDataObject or SmartBusinessObject) or an include file that provides data definitions for the SmartDataBrowser. Click Browse to select either a SmartDataObject, a SmartBusinessObject, or an Include file.</td>
</tr>
<tr>
<td>Help on definition source</td>
<td>Open the help on to definition source.</td>
</tr>
<tr>
<td><strong>SmartDataBrowser Wizard - Page 3 of 4</strong></td>
<td></td>
</tr>
<tr>
<td>Add Fields</td>
<td>Open the Multi-Field Selector dialog which allows you to add a list of fields from the selected DataObject to the SmartDataBrowser to make available to visualization objects.</td>
</tr>
<tr>
<td>Help on Fields</td>
<td>Open the help on the Fields dialog.</td>
</tr>
<tr>
<td><strong>SmartDataBrowser Wizard - Page 4 of 4</strong></td>
<td></td>
</tr>
<tr>
<td>Finish</td>
<td>Complete the wizard to create a SmartDataBrowser.</td>
</tr>
</tbody>
</table>
For more information, see Creating ABL GUI procedure files on page 644

**SmartDataViewer Wizard**

The SmartDataViewer Wizard allows you to create a SmartDataBrowser (SDV) object. The wizard allows you to first choose a DataObject to supply data to SmartDataViewer, and then you specify columns that will be displayed in the SmartDataBrowser object.

Access the SmartDataBrowser wizard by choosing SmartDataBrowser in the Object Type list on the Create an ABL GUI procedure dialog.

**SmartDataBrowser Wizard** is a four page wizard and includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SmartDataViewer Wizard - Page 1 of 4</strong></td>
<td></td>
</tr>
<tr>
<td>Next</td>
<td>Move to the next page of the wizard.</td>
</tr>
<tr>
<td><strong>SmartDataViewer Wizard - Page 2 of 5</strong></td>
<td></td>
</tr>
<tr>
<td>Data definition source</td>
<td>Specify a DataObject (SmartDataObject or SmartBusinessObject) or an include file that provides data definitions for the SmartDataBrowser. Click Browse to select either a SmartDataObject, a SmartBusinessObject, or an Include file.</td>
</tr>
<tr>
<td>Help on definition source</td>
<td>Open the help on definition source.</td>
</tr>
<tr>
<td><strong>SmartDataViewer Wizard - Page 3 of 4</strong></td>
<td></td>
</tr>
<tr>
<td>Add Fields</td>
<td>Open the Multi-Field Selector dialog which allows you to add a list of fields from the selected DataObject to the SmartDataBrowser to make available to visualization objects.</td>
</tr>
<tr>
<td>Help on Fields</td>
<td>Open the help on the Fields dialog.</td>
</tr>
<tr>
<td><strong>SmartDataViewer Wizard - Page 4 of 4</strong></td>
<td></td>
</tr>
<tr>
<td>Finish</td>
<td>Complete the wizard to create a SmartDataViewer.</td>
</tr>
</tbody>
</table>

For more information, see Creating ABL GUI procedure files on page 644

**Property Sheet for SmartObjects**

The Property Sheet dialog allows you to edit the attributes of a selected SmartObject instance.

This dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>Specify a name for the selected SmartObject instance or accept the default name.</td>
</tr>
<tr>
<td>Type</td>
<td>Displays the SmartObject type for the selected SmartObject instance.</td>
</tr>
</tbody>
</table>
## Option | Description
--- | ---
**Master File** | Displays the name of the master file from which the selected SmartObject instance is created.

**File** | Opens the *Modify Master File Name* dialog which allows you to change the name of the master file from which the selected SmartObject instance is created.

**Instance Properties** | Displays the instance properties for the selected SmartObject instance.

**Edit** | Allows you to modify the instance properties of the selected SmartObject instance.

**Parameterize as Variable** | Select this check box to use the selected SmartObject instance as a placeholder.

**Variable** | Specify a variable name to use when parameterizing the selected SmartObject instance. 

*Note:* The *Variable* field is only displayed when the *Parameterize as Variable* check box is selected.

**SmartLinks** | Opens the *SmartLinks* dialog which allows you to view, add, and remove SmartLinks between SmartObject instances in the current design window.

**SmartInfo** | Opens the *SmartInfo* dialog which allows you to view information on the SmartLinks supported by the selected SmartObject instance, as well as the database tables used and supplied by the SmartObject instance.

### See also
- Inserting SmartObjects on page 669
- Choose SmartObject dialog on page 788
- SmartObject context menu on page 694

## SmartObject Properties dialog

The *SmartObject Properties* dialog allows you to view the available properties and their corresponding values for the selected SmartObject.

Access this dialog by clicking *Properties* on the *SmartInfo* dialog.

The *SmartObject Properties* dialog includes the following options:

## Option | Description
--- | ---
**Property** | Displays the name of the property.

**Value** | Displays the current value of the property.

For more information, see Property Sheet for SmartObjects
**SmartDataObject Instance Properties dialog**

The **SmartDataObject Instance Properties** dialog allows you to view the available properties and their corresponding values for the currently selected SmartDataObject.

Access this dialog by choosing **Instance Properties** on the **SmartObject** right-click context menu.

The instance properties dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition</td>
<td>Select a partition name from the drop-down.</td>
</tr>
<tr>
<td>Read data in batches</td>
<td>Specify the number of rows in the <strong>Rows</strong> field, to be retrieved from the database and stored by the client. The default value is 200.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The <strong>Rows</strong> field will be enabled only when the <strong>Read data in batches</strong> check box is selected.</td>
</tr>
<tr>
<td>Rebuild dataset on Reposition</td>
<td>Select to rebuild the temp-table starting with the specified record.</td>
</tr>
<tr>
<td>Open query on initialization</td>
<td>Select to open the query when the SmartDataObject initializes. This check box is selected by default.</td>
</tr>
<tr>
<td>Instance name</td>
<td>The identifier of this object instance, used by SmartBusinessObjects and in similar contexts to distinguish one SmartDataObject from another. The default value is the filename root without the file path and extension. You can change it to some other value.</td>
</tr>
<tr>
<td>Check current changed</td>
<td>Select to check on update that records in the database have not been changed since they were retrieved by this instance of the SmartDataObject.</td>
</tr>
<tr>
<td>Activate/deactivate DataTargets on view/hide</td>
<td>Defines whether or not the SmartDataObject retrieves records needed by objects that are currently hidden.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> By default, the <strong>Activate/deactivate DataTargets on view/hide</strong> check box is selected.</td>
</tr>
<tr>
<td>Prompt on delete</td>
<td>Select to display confirmation prompt for record deletions.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> By default, the <strong>Prompt on delete</strong> check box is selected.</td>
</tr>
</tbody>
</table>
### Display fields for prompt

Select which field values display when prompting to confirm record deletions. Select any one of the following options:

- **None** – to display no field values.
- **All** – to display all field values.
- **Select Fields** – to display only the selected field values.

### Edit display field list

Opens the **Multi-Field Selector** dialog which allows you to select the fields. The selected field values are displayed for record deletion prompts.

**Note:** The **Edit display field list** button is enabled only when you choose **Select Fields** in the **Display fields for prompt** section.

---

**See also**

- [Inserting SmartObjects](#) on page 669
- [SmartObjects context menu](#) on page 694

**DataView Instance Properties dialog**

The **DataView Instance Properties** dialog allows you to configure the behavior of your DataView instance. Access this dialog by choosing **Instance Properties** on the DataView right-click context menu.

The **DataView Instance Properties** dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Business entity</strong></td>
<td>The logical name of the business entity on which the DataView operates.</td>
</tr>
<tr>
<td><strong>Note:</strong> This is the logical name. The physical business entity name must be realized by the service before the DataView can function.</td>
<td></td>
</tr>
<tr>
<td><strong>Read data in batches</strong></td>
<td>Specify the number of rows in the <strong>Rows</strong> field, to be retrieved from the database and stored by the client. The default value is 200.</td>
</tr>
<tr>
<td><strong>Note:</strong> The <strong>Rows</strong> field will be enabled only when the <strong>Read data in batches</strong> check box is selected.</td>
<td></td>
</tr>
<tr>
<td><strong>Rebuild dataset on Reposition</strong></td>
<td>Select to rebuild the temp-table starting with the specified record.</td>
</tr>
<tr>
<td><strong>Open query on initialization</strong></td>
<td>Select to open the query when the DataView object initializes. This check box is selected by default.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Instance name</td>
<td>The instance name of the DataSet. The default value is the business entity name. Specifying a different value enables separate instances of the same business entity to run in a container.</td>
</tr>
<tr>
<td>Data table</td>
<td>Select a data table in the business entity. You can edit and navigate through the selected table.</td>
</tr>
<tr>
<td>View</td>
<td>Lists any other tables associated with the selected table. For each listed table, you can set the following options:</td>
</tr>
<tr>
<td></td>
<td>• View – Determines if data from this table is available for display by visual objects linked to the data table. Only tables that are directly linked to the data table can be part of the data table’s view. The table’s records must also be uniquely identifiable from the data table.</td>
</tr>
<tr>
<td></td>
<td>• Join – Determines if this table can be browsed, sorted, or filtered. Only tables that are inner-joined with the data table should be selected.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The parent table in the DataSet cannot be included through the Join option.</td>
</tr>
<tr>
<td>Sort by</td>
<td>Lists the fields used to sort the results.</td>
</tr>
<tr>
<td>Edit Sort</td>
<td>Enables you to add fields to the Sort by list and arrange their order.</td>
</tr>
<tr>
<td>Ascending/Descending</td>
<td>Enables you to specify the sort order for the currently selected field in the Sort by list.</td>
</tr>
<tr>
<td>Inst. name</td>
<td>The name of the DataView instance.</td>
</tr>
<tr>
<td>Activate/deactivate DataTargets on view/hide</td>
<td>Defines whether or not the DataView retrieves records needed by objects that are currently hidden.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> By default, the Activate/deactivate DataTargets on view/hide check box is selected.</td>
</tr>
<tr>
<td>Prompt on delete</td>
<td>Select to display confirmation prompt for record deletions.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> By default, the Prompt on delete check box is selected.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Display fields for prompt | Select which field values display when prompting to confirm record deletions. Select any one of the following options:  
  • None – to display no field values.  
  • All – to display all field values.  
  • Select Fields – to display only the selected field values. |
| Edit display field list    | Opens the Multi-Field Selector dialog which allows you to select the fields. The selected field values are displayed for record deletion prompts.  
  Note: The Edit display field list button is enabled only when you choose Select Fields in the Display fields for prompt section. |

**See also**

Inserting SmartObjects on page 669  
SmartObjects context menu on page 694

**SmartFolder Instance Properties dialog**

The SmartFolder Instance Properties dialog allows you to modify the attributes of a selected SmartFolder instance.

Access this dialog by choosing Instance Properties on the SmartFolder right-click context menu.

The SmartFolder Instance Properties dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tab Labels</td>
<td>Displays the tab labels. If a tab label is left blank, no tab is generated for that position.</td>
</tr>
<tr>
<td>Insert</td>
<td>Add a label for the tabs on the SmartFolder.</td>
</tr>
<tr>
<td>Remove</td>
<td>Delete a selected tab label.</td>
</tr>
<tr>
<td>Use Fixed Tabs</td>
<td>Specify the width of the tabs on the SmartFolder instance. The width of the tabs can be either 110 pixels or 72 pixels per tab. The default value is 110 pixels per tab.</td>
</tr>
<tr>
<td>Font</td>
<td>Opens the Choose Folder Font dialog which allows you to select a font for the text appearing on the SmartFolder.</td>
</tr>
</tbody>
</table>

**See also**

Inserting SmartObjects on page 669  
SmartObjects context menu on page 694
SmartPanel Instance Properties dialog

The SmartPanel Instance Properties dialog allows you to configure the behavior of SmartPanel instances. Access this dialog by choosing Instance Properties on the SmartPanel instance right-click context menu.

The instance properties dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Show Border</td>
<td>Determines whether the decorative rectangle is shown. The default is &quot;YES.&quot; If set to &quot;NO,&quot; the Edge Pixels is forced to &quot;0.&quot;</td>
</tr>
<tr>
<td>Edge Pixels</td>
<td>Sets the EDGE-PIXELS attribute which determines whether the SmartPanel has a decorative rectangle and, if so, the number of pixels used as an edge width. The default is &quot;2.&quot; Setting this to &quot;0&quot; has the effect of removing the rectangle.</td>
</tr>
<tr>
<td>TableIO SmartPanel Type</td>
<td>Set the SmartPanel Type attribute which determines which mode (save or update) the update panel is in. By default, this is set to &quot;Save.&quot; (This attribute is supported for the Standard Update panel only).</td>
</tr>
<tr>
<td>Behavior of Add Button</td>
<td>Set the AddFunction attribute which determines whether the SmartPanel Add button prompts you for a single new record or multiple new records. By default, this is set to &quot;One-Record.&quot; You can also set it to &quot;Multiple-Records.&quot; (This attribute is supported for the Standard Update panel only).</td>
</tr>
</tbody>
</table>

See also

Inserting SmartObjects on page 669
SmartObjects context menu on page 694

SmartDataViewer Instance Properties dialog

SmartDataViewer Instance Properties dialog

The SmartDataViewer Instance Properties dialog allows you to modify the attributes and layout of a selected SmartDataViewer instance when it is created at run time. You can also specify which non-database objects with DataFields you want to disable.

Access this dialog by choosing Instance Properties on the SmartFolder right-click context menu.

The instance properties dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Name</td>
<td>Displays the name of the Viewer instance.</td>
</tr>
<tr>
<td>Behavior during Initialize</td>
<td></td>
</tr>
<tr>
<td>Enable</td>
<td>Determines whether the Viewer instance is initially shown when it is created at run time.</td>
</tr>
<tr>
<td>View</td>
<td>Determines whether the Viewer instance is initially shown when it is created at run time. This toggle box sets the Hide-on-Init attribute to YES or NO. The default setting is YES, meaning that the object is initially visible when it is created at run time.</td>
</tr>
</tbody>
</table>
If you defined multiple layouts for the Viewer master, let's you select which layout you want to use for this instance. Sets the apply-layout instance attribute.

This section only applies to non-database objects that you have defined as Enabled in the master object.

Disables all non-database objects with DataFields.

Leaves all non-database objects enabled.

Lets you select which non-database objects you want to disable with DataFields. When you select this option, you can then check which non-database objects you want to disable.

This section lets you specify which fields to use to set DataModified.

Let you select the fields that can set DataModified. When you select this option, you can then check which fields you want used to set DataModified.

All enabled fields should set DataModified. This is the default option.

Only fields that are enabled in the data source should set DataModified.

Only enabled fields that can be updated in the data source should set DataModified.

No fields should set DataModified.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layout</td>
<td>If you defined multiple layouts for the Viewer master, let’s you select which layout you want to use for this instance. Sets the apply-layout instance attribute.</td>
</tr>
<tr>
<td>Which objects should be disabled with data fields?</td>
<td>This section only applies to non-database objects that you have defined as Enabled in the master object.</td>
</tr>
<tr>
<td>Some</td>
<td>Disables all non-database objects with DataFields.</td>
</tr>
<tr>
<td>All</td>
<td>Leaves all non-database objects enabled.</td>
</tr>
<tr>
<td>None</td>
<td>Lets you select which non-database objects you want to disable with DataFields. When you select this option, you can then check which non-database objects you want to disable.</td>
</tr>
<tr>
<td>Which fields should set DataModified on change?</td>
<td>This section lets you specify which fields to use to set DataModified.</td>
</tr>
<tr>
<td>Some</td>
<td>Let you select the fields that can set DataModified. When you select this option, you can then check which fields you want used to set DataModified.</td>
</tr>
<tr>
<td>All</td>
<td>All enabled fields should set DataModified. This is the default option.</td>
</tr>
<tr>
<td>Data Fields</td>
<td>Only fields that are enabled in the data source should set DataModified.</td>
</tr>
<tr>
<td>Updatable</td>
<td>Only enabled fields that can be updated in the data source should set DataModified.</td>
</tr>
<tr>
<td>None</td>
<td>No fields should set DataModified.</td>
</tr>
</tbody>
</table>

See also

- Inserting SmartObjects on page 669
- SmartObjects context menu on page 694

**SmartFilter Instance Properties dialog**

The SmartDataObject Instance Properties dialog allows you to select the fields from the filter target to display in the SmartFilter, and to specify field properties, style, and appearance.

Access this dialog by choosing Instance Properties on the SmartFilter right-click context menu.

The instance properties dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Section</td>
<td>Displays the name of the filter target (a SmartDataObject or a SmartDataBrowser based on a database).</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Fields</td>
<td>Displays a list of the selected fields in the filter target in the order they will be displayed in the SmartFilter.</td>
</tr>
<tr>
<td>Edit Field List</td>
<td>Opens the <strong>Multi-Field Selector</strong> dialog to select, deselect, or rearrange fields.</td>
</tr>
<tr>
<td>Remove</td>
<td>Deletes the selected field from list of available fields without starting up the Multi-Field selector.</td>
</tr>
<tr>
<td><strong>Field Properties Section</strong></td>
<td></td>
</tr>
<tr>
<td>Label</td>
<td>Displays label of the selected field.</td>
</tr>
<tr>
<td>Filter Target</td>
<td>When checked, the label used is the same as that used in the filter target. When unchecked, you can specify a different label for the selected field.</td>
</tr>
<tr>
<td>Width</td>
<td>Displays width of the selected field.</td>
</tr>
<tr>
<td>Use Default</td>
<td>When checked the width field displays the default width specified in the filter class. When unchecked, you can specify a different width.</td>
</tr>
<tr>
<td>ToolTip</td>
<td>Allows you to enter text for tool tip help.</td>
</tr>
<tr>
<td>Help ID</td>
<td>Enter an identifier for the Help topic associated with this field.</td>
</tr>
<tr>
<td>View as Range Fields</td>
<td>Use to override the style specified for all fields in the Style section. Check to specify that the selected field will have the same style as the Range specification would give to all fields.</td>
</tr>
<tr>
<td>Explicit Operator</td>
<td>Use to override the style specified for all fields in the Style section. Check to specify that the selected field will have the same style as the Explicit specification would give to all fields.</td>
</tr>
<tr>
<td><strong>Style Section</strong></td>
<td></td>
</tr>
<tr>
<td>Implicit</td>
<td>Select to generate a single list of fields of correct data type, into which the user can enter filter data. Specify the implicit operator by selecting it from the combo box to the right.</td>
</tr>
<tr>
<td>Explicit</td>
<td>Select to display a selection of operators with the filter field, so that the user can choose an explicit operator at run time. (The visualization of the display is determined in the Operator View as section below.)</td>
</tr>
<tr>
<td>Range</td>
<td>Select to allow the user to specify a range of inclusive values for each field.</td>
</tr>
<tr>
<td>Inline</td>
<td>Select to generate a single list of fields into which the user can enter the operator &quot;inline&quot; with the filter data. All fields are character fields to allow the typing of the operator. If no operator is specified, the default is EQUALS, unless BEGINS is specified in the String Operator section below.</td>
</tr>
<tr>
<td><strong>String Operator Section</strong></td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>BEGINS</td>
<td>Select to specify that filter data in character fields are evaluated with BEGINS. This overrides options specified in the Implicit and Inline styles.</td>
</tr>
<tr>
<td>Operator View as Section</td>
<td></td>
</tr>
<tr>
<td>Combo-box</td>
<td>Select to specify that the explicit operator is visualized as a combo-box.</td>
</tr>
<tr>
<td>Radio-set</td>
<td>Select to specify that the explicit operator is visualized as a radio-set.</td>
</tr>
</tbody>
</table>

**See also**

- Inserting SmartObjects on page 669
- SmartObjects context menu on page 694

**SmartBusinessObject Instance Properties dialog**

The **SmartBusinessObject Instance Properties** dialog allows you to modify the attributes of a SmartBusinessObject instance.

Access this dialog by choosing **Instance Properties** on the SmartBusinessObject instance right-click context menu.

The instance properties dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition</td>
<td>Select the AppServer partition to be associated with this instance. If there are no partitions defined, use the Service Parameter Maintenance PRO*Tool to define one or more.</td>
</tr>
<tr>
<td>Cascade on Browse</td>
<td>By default, this check box is selected. Clearing this check box will cause navigation changes to propagate only until more than one record is returned in a dataset.</td>
</tr>
<tr>
<td>Place SDOs in order</td>
<td>You must arrange the list of SmartDataObjects to reflect the synchronization hierarchy. The synchronization hierarchy is defined by the direction and order of the DATA SmartLinks you use to connect the objects. Use the Move Up/Down buttons to change the position of the current item. SmartDataObjects whose DATA links come from the same source are on the same level, and their order here is not significant.</td>
</tr>
</tbody>
</table>

**See also**

- Inserting SmartObjects on page 669
- SmartObjects context menu on page 694

**SmartProducer Instance Properties dialog**

The **SmartProducer Instance Properties** dialog allows you to configure the behavior of your SmartProducer instances.

Access this dialog by choosing **Instance Properties** on the SmartProducer instance right-click context menu.
The instance properties dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>JMS Session section</strong></td>
<td></td>
</tr>
<tr>
<td>Domain</td>
<td>Defaults to Publish-and-Subscribe. Choosing Point-to-Point causes the message to be deleted from the queue by the first entity reading it. In contrast, every subscriber can read every message under the Publish-and-Subscribe system.</td>
</tr>
<tr>
<td>JMS Partition</td>
<td>Select a partition from the list. If there are none defined, create at least one using the Service Parameters Maintenance PRO*Tool.</td>
</tr>
<tr>
<td>Ping Interval</td>
<td>Defaults to 0 (no pinging). You can enter an integer value representing seconds here. If the value is non-zero, the SmartProducer will ping the SonicMQ Broker at that interval throughout the session.</td>
</tr>
<tr>
<td>JMS User</td>
<td>Type the user identifier to be used at login. This is a mandatory field if security is on.</td>
</tr>
<tr>
<td>JMS Password</td>
<td>Type the password of the JMS user. This is a mandatory field if security is on.</td>
</tr>
<tr>
<td>JMS Client</td>
<td>Type identifier unique to this client. This is a mandatory field if security is on, and the User and Password values are not unique to this client.</td>
</tr>
<tr>
<td>Prompt for JMS Login</td>
<td>Select this check box to cause the human user to be prompted at startup. This is a mandatory field when security is on.</td>
</tr>
<tr>
<td><strong>Message Delivery Defaults section</strong></td>
<td></td>
</tr>
<tr>
<td>Priority</td>
<td>Defaults to 4. Read whenever a new message is sent. Set to some value in the range 0 (lowest) to 9 (highest). Modifiable under program control.</td>
</tr>
<tr>
<td>Time to Live</td>
<td>Defaults to 0 (no expiration). Set to some value in milliseconds (1000 milliseconds = 1 second). Determines how long a message can remain unread without being discarded as stale. Modifiable under program control.</td>
</tr>
<tr>
<td>Persistency</td>
<td>Defaults to PERSISTENT (message stored in broker’s database). Other values supported are NON_PERSISTENT (message not stored),</td>
</tr>
<tr>
<td></td>
<td>NON_PERSYSTENT_ASYNC, and UNKNOWN.</td>
</tr>
<tr>
<td>Message Type</td>
<td>Defaults to TextMessage. Other types supported are BytesMessage, HeaderMessage, XMLMessage (do not use with SmartB2BObject), MapMessage, StreamMessage.</td>
</tr>
</tbody>
</table>

**See also**

- Inserting SmartObjects on page 669
- SmartObjects context menu on page 694
SmartConsumer Instance Properties dialog

The **SmartConsumer Instance Properties** dialog allows you to configure the behavior of SmartConsumer instances.

Access this dialog by choosing **Instance Properties** on the SmartConsumer instance right-click context menu.

The instance properties dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>JMS Session section</strong></td>
<td></td>
</tr>
<tr>
<td>Publish and Subscribe</td>
<td>Defaults to Publish-and_Subscribe. Choosing Point-to-Point causes the message to be deleted from the queue by the first entity reading it. In contrast, every subscriber can read every message under the Publish-and-Subscribe system.</td>
</tr>
<tr>
<td>JMS Partition</td>
<td>Select a a partition from the list. If there are none defined, create at least one using the Service Parameters Maintenance PRO*Tool.</td>
</tr>
<tr>
<td>Ping Interval</td>
<td>Defaults to 0 (no pinging). You can enter an integer value representing seconds here. If the value is non-zero, the SmartProducer will ping the SonicMQ Broker at that interval throughout the session.</td>
</tr>
<tr>
<td>Log File</td>
<td>Defaults to empty. Set this field to the pathname of the log file to be used in batch mode.</td>
</tr>
<tr>
<td>Shutdown</td>
<td>Defaults to empty. Set this field to the identifier of the Topic/Queue to be used for shutting down this object in batch mode.</td>
</tr>
<tr>
<td>JMS User</td>
<td>Type the user identifier to be used at login. This is a mandatory field if security is on.</td>
</tr>
<tr>
<td>JMS Password</td>
<td>Type the password of the JMS user. This is a mandatory field if security is on.</td>
</tr>
<tr>
<td>JMS Client</td>
<td>Type identifier unique to this client. This is a mandatory field if security is on, and the User and Password values are not unique to this client.</td>
</tr>
<tr>
<td>Prompt for JMS Login</td>
<td>Select this check box to cause the human user to be prompted at startup. This is a mandatory field when security is on.</td>
</tr>
<tr>
<td><strong>Destinations section</strong></td>
<td></td>
</tr>
<tr>
<td>Destination (list)</td>
<td>Lists the destinations this object will monitor for traffic.</td>
</tr>
<tr>
<td>Destination (fill-in)</td>
<td>Set to the identifier of the Topic/Queue to be monitored.</td>
</tr>
<tr>
<td>Message Selector</td>
<td>Defaults to empty. Enter the boolean expression (for example: ‘priority &lt; 5’) to be used when filtering incoming messages. Messages that fail the test will be discarded without notice.</td>
</tr>
<tr>
<td>Durable Subscription</td>
<td>Select this check box will cause the subscription to persist across sessions.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Subscription Name</td>
<td>Type in an identifier for this durable subscription.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This is enabled only when the Durable Subscription check box is selected.</td>
</tr>
<tr>
<td>Unsubscribe on Close</td>
<td>Select this check box to cancel the subscription at the end of the session.</td>
</tr>
</tbody>
</table>

**See also**
- Inserting SmartObjects on page 669
- SmartObjects context menu on page 694

### SmartRouter Instance Properties dialog

The **SmartRouter Instance Properties** dialog allows you to configure the behavior of SmartRouter instances. Access this dialog by choosing **Instance Properties** on the SmartRouter instance right-click context menu.

The instance properties dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>External Reference</td>
<td>Enter the URL of the schema file for a document type that this SmartRouter instance must recognize. Note that the actual filename must be the same as the filename of the local copy, though the rest of the path will almost certainly be different.</td>
</tr>
<tr>
<td>Internal Reference</td>
<td>Enter the PROPATH-relative path to the map file for this document type.</td>
</tr>
</tbody>
</table>

**See also**
- Inserting SmartObjects on page 669
- SmartObjects context menu on page 694

### SmartSender Instance Properties dialog

The **SmartSender Instance Properties** dialog allows you to configure the behavior of SmartRouter instances. Access this dialog by choosing **Instance Properties** on the SmartSender instance right-click context menu.

The instance properties dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination</td>
<td>Enter the identifier of the SonicMQ Topic or Queue to which messages created by this object will be sent.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Reply Requested</td>
<td>Defaults to clear. Select this check box to indicate message recipients to be notified that you want a reply.</td>
</tr>
<tr>
<td>Reply Selector</td>
<td>Defaults to empty. You can enter a boolean expression here (for example &quot;priority &lt;= 5&quot;) that refers to reply-header fields. Every reply will be tested against that expression and replies failing the test will be discarded without processing or notice.</td>
</tr>
</tbody>
</table>

**See also**
- Inserting SmartObjects on page 669
- SmartObjects context menu on page 694

**Visual SmartObject dialog**

The Visual SmartObject dialog allows you to specify certain Visual SmartObject instance properties.

Access this dialog by choosing Instance Properties on the SmartObject instance right-click context menu.

The instance properties dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>Determines whether the SmartObject instance is initially disabled when it is created at run time. This toggle box sets the Disable-on-Init attribute to YES or NO. The default setting is NO, meaning that the SmartObject is initially enabled when it is created at run time.</td>
</tr>
<tr>
<td>View</td>
<td>Determines whether the SmartObject instance is initially hidden when it is created at run time. This toggle box sets the Hide-on-Init attribute to YES or NO. The default setting is NO, meaning that the SmartObject is initially visible when it is created at run time.</td>
</tr>
<tr>
<td>Layout</td>
<td>The names of any alternate layouts you have created in the SmartObject’s master file appear in this combo-box, along with the [default] option. The AppBuilder sets the Layout property to the value you select. The [default] option does not set the Layout attribute to any value.</td>
</tr>
</tbody>
</table>

**See also**
- Inserting SmartObjects on page 669
- SmartObjects context menu on page 694

**SmartInfo dialog**

The SmartInfo dialog allows you to view SmartLink information about the current SmartObject.

A SmartLink establishes how one SmartObject relates to another SmartObject and what sort of behavior it can expect from the other SmartObject. A SmartLink is bi-directional association of two SmartObjects.

Access this dialog by clicking the SmartInfo icon on the Property Sheet -SmartObject instance dialog.

The SmartInfo dialog includes the following options:
### Option Description

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master File</td>
<td>Displays the master file name of the selected SmartObject instance.</td>
</tr>
<tr>
<td>Type</td>
<td>Displays the SmartObject type for the selected SmartObject instance.</td>
</tr>
<tr>
<td>ADM Supported Links</td>
<td>Displays the SmartLinks supported by the ADM for the current SmartObject.</td>
</tr>
<tr>
<td>Requires Record-Source for these Tables</td>
<td>Displays a list of tables for which the current SmartObject requires a RECORD-SOURCE.</td>
</tr>
<tr>
<td>Can Send Records in these Tables</td>
<td>Displays a list of tables for which the current SmartObject can act as RECORD-SOURCE.</td>
</tr>
<tr>
<td>Properties</td>
<td>Displays the <a href="https://docs.progress.com/edge/developers/developer-studio/doc">SmartObject Properties</a> dialog.</td>
</tr>
</tbody>
</table>

For more information, see [Property Sheet for SmartObjects](https://docs.progress.com/edge/developers/developer-studio/doc).

### SmartLinks dialog

The **SmartLinks** dialog allows you to view, add, modify, and remove SmartLinks between SmartObjects in the current design window.

Access this dialog by clicking **SmartLinks** icon on the **Property Sheet -SmartObject instance** dialog.

The **SmartLinks** dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Displays the source object for the link relationship.</td>
</tr>
<tr>
<td>Link Type</td>
<td>Displays the type of link relationship.</td>
</tr>
<tr>
<td>Target</td>
<td>Displays the target object for the link relationship.</td>
</tr>
<tr>
<td>Filters</td>
<td>Provides the filter options such as source object, link type, and target object.</td>
</tr>
<tr>
<td>Show</td>
<td>Select any of the following options:</td>
</tr>
<tr>
<td>Sort Order</td>
<td>Select any of the following options:</td>
</tr>
</tbody>
</table>

- **All Links** – to display all SmartLinks in the design window.
- **To/From** – to display only those SmartLinks that have a specific SmartObject as a source or target. Use the drop-down to select a SmartObject.

**Note:** The SmartObject drop-down is enabled only when the **To/From** option is selected.

- **Source** – to sort the SmartLinks on the design window based on source objects.
- **Link** – to sort the SmartLinks on the design window based on the link types.
- **Target** – to sort the SmartLinks on the design window based on the target objects.
Open the Add a SmartLink dialog which allows you to add a SmartLink between two SmartObjects in the current design window.

Open the Modify a SmartLink dialog which allows you to edit the currently selected link.

Deletes a selected SmartLink.

Validates the links added on the currently design window.

For more information, see Property Sheet for SmartObjects

**Add a SmartLink dialog**

The Add a SmartLink dialog allows you to add a SmartLink between two SmartObjects. Access this dialog by clicking Add on the SmartLinks dialog.

The Add a SmartLink dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Displays all of the SmartObjects in the current design window.</td>
</tr>
<tr>
<td>Link Type</td>
<td>Displays the allowed link types for the source object selected in the Source column.</td>
</tr>
<tr>
<td>Target</td>
<td>Displays all of the SmartObjects in the current design window.</td>
</tr>
<tr>
<td>Info on Source</td>
<td>Displays SmartInfo for the selected source object.</td>
</tr>
<tr>
<td>Info on Target</td>
<td>Displays SmartInfo for the selected target object.</td>
</tr>
</tbody>
</table>

For more information, see SmartLinks dialog

**Modify a SmartLink dialog**

The Modify a SmartLink dialog allows you to modify the SmartLink between two SmartObjects. Access this dialog by clicking Modify on the SmartLinks dialog.

The Modify a SmartLink dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Displays all of the SmartObjects in the current design window.</td>
</tr>
<tr>
<td>Link Type</td>
<td>Displays the allowed link types for the source object selected in the Source column.</td>
</tr>
<tr>
<td>Target</td>
<td>Displays all of the SmartObjects in the current design window.</td>
</tr>
<tr>
<td>Info on Source</td>
<td>Displays SmartInfo for the selected source object.</td>
</tr>
<tr>
<td>Info on Target</td>
<td>Displays SmartInfo for the selected target object.</td>
</tr>
</tbody>
</table>
For more information, see SmartLinks dialog

Modify Master File Name dialog

The Modify Master File Name dialog allows you to select the master file for the selected SmartObject instance. Access by dialog by clicking File on the Property Sheet - SmartObject instance dialog.

The Modify Master File Name dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name</td>
<td>Specify the name of the file to use as the master file for the selected SmartObject instance.</td>
</tr>
<tr>
<td>Files</td>
<td>Choose to browse directories and select a master file.</td>
</tr>
</tbody>
</table>

For more information, see Property Sheet for SmartObjects

Help on Fields dialog

Choose Add Fields to display a dialog where you can select which fields from the SmartDataObject or database you want to include in your report. Note that you might not want to display all the fields that are part of your query or the SmartDataObject.

See also

SmartDataBrowser Wizard on page 790
SmartDataViewer Wizard on page 791

AppBuilder Queries

A set of instructions used to find the contents of one or more tables in your database. Creating a query consists of selecting the following:

- Tables that comprise the query
- Criteria you want to use to process the information in the query
- SmartV8Query

A SmartV8Query is a Version 8 SmartObject that defines a set of records to fetch from a database. A SmartV8Query usually has no run-time visualization (unless you specifically add a frame to it).

For more information, see SmartDataObject Wizard

Help on DataObject

A SmartDataObject is a Version 9 SmartObject that holds a query and associated data update logic. It has at its back end a database query and at its front end an open interface to visual objects.

It can serve as a data source and a data target for other SmartObjects, such as SmartDataBrowsers and SmartDataViewers. A SmartDataObject performs update for its related visual objects and can provide validation.

SmartDataBrowser is a Version 9 SmartObject that connects either to the database or to a SmartDataObject and holds a browse object to view database records and to update them through the SmartDataObject.

SmartDataViewer is a Version 9 SmartObject that holds a frame to preview and possibly update specific database fields. Database field values are obtained from a SmartDataObject or a SmartDataBrowser.
See also
SmartDataObject Wizard on page 788
SmartDataBrowser Wizard on page 790
SmartDataViewer Wizard on page 791

Preview SmartObject dialog box

Use the Preview SmartObject dialog box to view a SmartObject.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master File</td>
<td>Displays the pathname of the SmartObject master file (a SmartObject with specific data and UI) for the selected SmartObject.</td>
</tr>
<tr>
<td>Type</td>
<td>Displays the type of the selected SmartObject.</td>
</tr>
</tbody>
</table>

See also
About SmartObjects on page 637
Open SmartObject dialog box on page 788

Progress Dynamic dialogs

Open Object dialog

The Open Object dialog enables you to browse for objects stored in the Repository and open them for editing in the GUI Designer.

You can access this dialog by selecting the Open Object option on the File menu.

Note: The Open Dialog menu option is available only with the Dynamic projects. See Setting Progress Dynamics project.

The Open Object dialog includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object filename</td>
<td>Enter a full or partial object file name to automatically filter the objects listed in the Open Object dialog. As you enter each character, the list of objects in the browser narrows. Select an object in the browser to open it or select it for launch from the Dynamic Launcher dialog box.</td>
</tr>
<tr>
<td>Module</td>
<td>Select a product module to display objects created in that product module.</td>
</tr>
<tr>
<td>Type</td>
<td>Select an object type to display objects of that type.</td>
</tr>
</tbody>
</table>
**Object browser**
Displays the objects stored in the Repository for the selected product module or type, or that match the filename filter value.

**Note:** You can change how the browse displays information in each column by clicking on the column name. For example, if you click on the **Object filename** column, all columns are sorted and displayed by the object filename in alphabetical order. If you click on **Object type code**, all columns are sorted and displayed by object type code in alphabetical order. You can use any column to sort and display the information in the browse.

Right-click an object to access the following context menu options:

- **Open** - Opens the selected object for editing or selects the object for launch from the Dynamic Launcher dialog box
- **Remove from Repository** - Displays an alert box to confirm the removal of the selected object from the Repository.
- **Properties** - Displays the **Object Properties** dialog which allows you to view properties for the selected object.

<table>
<thead>
<tr>
<th>Object filename</th>
<th>Lists the most recently opened object list. To open an object, select the object and click Open.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Open</strong></td>
<td>Opens the selected object for editing, or selects the object for launch from the Dynamic Launcher dialog.</td>
</tr>
</tbody>
</table>

**See also**
- Opening dynamics object on page 664
- Editing and saving dynamics object on page 665
- Progress Dynamics menu options on page 690
- Save As Dynamic Object dialog on page 809

**Save As Dynamic Object dialog**
The **Save As Dynamic Object** dialog enables you to save the following static objects as dynamic objects: Static SmartDataViewers (StaticSDV), Static SmartDataBrowsers (StaticSDB), and Static SmartDataObjects (SDO).

**Note:** If you have a StaticDataViewer (SDV) that contains a SmartDataField (SDF) and you want to save the SDV as a dynamic object, you must first register the SDF in the ICFDB Repository. If you register the SDF before you save the static viewer as a dynamic viewer, the SDF is referenced properly by the new dynamic viewer. However, if the SDF is not registered before you try to migrate the viewer from static to dynamic, you get an error message requesting that you register the SDF and the dynamic version of the viewer is not created.

Access the **Save As Dynamic Object** dialog by selecting **File > Save Static Object As Dynamic**.
This dialog includes the following options:

| Dynamic Object Type | Displays the object type of the static object you have selected. This is a read-only field. |
Object Name | Displays the default name of static object you have selected. You can enter a different name to save the static object as dynamic.

Note: This must be a unique object name in the Repository.

Description | Enter a description of the object.

Product Module | Select the product module in which you want to save the object as dynamic.

Relative Directory | Displays the PROPATH-relative directory of the selected product module.

Create Custom Super Procedure | Select this check box to create a custom super procedure for the object.

Note: The default procedure name is displayed in the **File Name** field.

Root Directory | Displays the root directory of the custom super procedure. Click **Browse** to select a different directory.

Note: By default, your OpenEdge working directory displays as the root directory.

File Name | Displays the default file name of the custom super procedure. You can enter a different file name.

Type | Displays the type of the custom super procedure, as Procedure.

Product Module | Select the product module in which you want to save the custom super procedure file.

Relative Directory | Displays the PROPATH-relative directory of the selected product module.

Full Path Name | Displays the full path name of where the custom super procedure is stored. This is a read-only field.

---

**See also**

Saving static objects as dynamic on page 667
Saving dynamic objects as static on page 665
Progress Dynamics menu options on page 690

**Save As Static Object dialog**

The **Save As Static Object** dialog enables you to save the following dynamic objects as static: Dynamic SmartDataViewers (DynView) and Dynamic SmartDataObjects (DynSDO).

Access the **Save As Static Object** dialog by selecting **File > Save Dynamic Object As Static**.

This dialog includes the following options:

| File name | Displays the default name of the dynamic object you have selected. You can enter a different name. |
| **Save in** | Displays the default directory where the selected dynamic object is saved as static. You can click the **Browse** icon to select a different path. |
| **Register object** | Select this check box to register the dynamic object as static to the Repository. |
| **Object name** | Displays the name of the object, as it appears in the **File name** field. This is a read-only field. |
| **Description** | Displays a default description of the selected object. You can change the description. |
| **Root directory** | Displays the default root directory in which the dynamic object is saved as static. You can click **Browse** to select a different directory. |
| **Product module** | Select the product module in which you want to save the object. |
| **Full path name** | Displays the full path name of the directory in which the dynamic object is saved as static. This is a read-only field. |
| **Deploy to web** | Select this check box to deploy the object to web (for Web and client/server deployment types). |
| **Deploy to client** | Select this check box to deploy the object to the client machines (for WebClient and client/server deployment types). |
| **Deploy to server** | Select this check box to deploy the object to remote servers (for client/server, AppServer, and Web deployment types). |
| **Design object** | Select this check box to deploy the object only for design (development) purpose. |

**See also**

- **Saving dynamic objects as static** on page 665
- **Saving static objects as dynamic** on page 667
- **Progress Dynamics menu options** on page 690

**Save As Dynamic Object dialog**

The **Save As Dynamic Object** dialog enables you to save the following static objects as dynamic: SmartDataViewers, SmartDataBrowsers, and SmartDataObjects.

You can access this dialog by selecting **Save Static Object As Dynamic** from the **File** menu.

This dialog includes the following controls:

| **Dynamic Object Type** | Displays the object type of the object. This is a read-only field. |
| **Object Name** | Enter the object name. This must be a unique object name in the Repository. |

**Note:** By default, the object name as it appears in the Repository, is displayed in the field.
Description | Enter a description of the object.
---|---
Product Module | Select a product module from the list where you want to save the object.
Relative Directory | Displays the PROPATH-relative directory for the selected product module.
Create Custom Super Procedure | Select this check box to automatically create the custom super procedure for the object. The default procedure name displays in the **File Name** field.
Root Directory | Displays the default root directory for the custom super procedure. Click **Browse** to select a different root directory.
File Name | Displays the default custom super procedure file name. You can enter a different file name.
Type | Displays the type of custom super procedure. By default, the type if a Procedure.
Product Module | Select a product module from the list where you want to save the custom super procedure.
Relative Directory | Displays the PROPATH-relative directory for the selected product module.
Full Path Name | Displays the complete location where the custom super procedure is stored. This is a read-only field.

**Note:** This defaults to either the AB_source_code_directory session parameter value, if defined, or the working directory.

**See also**
- Saving static objects as dynamic on page 667
- Progress Dynamics menu options on page 690

**Dynamic Launcher dialog**

The **Dynamic Launcher** dialog box enables you to run a dynamic object from the Object Repository (after you save it in the Repository).

Access this dialog box from the main menu by clicking **Tools > Dynamic Launcher**.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name of container to launch</td>
<td>Specifies the name of the dynamic object to run. Either select a recently launched object from the selection list, or choose the Lookup button to open the Get Object dialog box from which you can locate and select an object to run. Next to the Lookup button is a Clear MRU button that you can use to clear the list of most recently used files.</td>
</tr>
<tr>
<td>Run persistent</td>
<td>This option runs the object instance as a persistent procedure. This check box is selected by default.</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Clear repository cache</td>
<td>This option clears the client-side cache to run the latest version of the object. This check box is selected by default.</td>
</tr>
<tr>
<td>Lookup</td>
<td>Opens the Get Object dialog box. Use this dialog box to select the object file you want to launch.</td>
</tr>
<tr>
<td>Clear MRU</td>
<td>Use this button to clear your list of most recently used files.</td>
</tr>
<tr>
<td>Destroy ADM Super Procedures</td>
<td>Deletes the current instance of any standard super procedures (that run in the client session) to run the latest version of the procedures.</td>
</tr>
<tr>
<td>Run</td>
<td>Runs the specified object.</td>
</tr>
<tr>
<td>Stop</td>
<td>Stops the launch process.</td>
</tr>
</tbody>
</table>

**See also**

Progress Dynamics menu options on page 690

**Views**

**Outline view**

**Toolbar buttons**

The Outline view toolbar buttons are available only when you have an AppBuilder procedure file open in view source mode.

The toolbar contains the following buttons:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Section Editor Toggle]</td>
<td>Switch to the Section Editor view. See Toggling to the Section Editor mode on page 658.</td>
</tr>
<tr>
<td>![Collapse All]</td>
<td>Collapse all expanded tree nodes in the Outline view.</td>
</tr>
<tr>
<td>![Expand Include File Contents]</td>
<td>Display elements in include files.</td>
</tr>
<tr>
<td>Control</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><img src="Image" alt="Filters" /></td>
<td>Display only selected element types in the Outline view. See Using Filter Preferences.</td>
</tr>
</tbody>
</table>
| ![Link with Editor](Image) | Enable this option to move to a declaration in the current file. Turn this option on or off by selecting (or clearing) it in the drop-down menu that appears when you click ![Icon](Image) icon on the Outline view toolbar.  
**Note:** To navigate to a specific place in the code from the Outline view, you must enable the **Link with Editor** option. |

**See also**
- Using the Outline view on page 657
- Outline view context menu on page 814

**Context menu**
Right-clicking a node in the **Outline** view displays the context menu. Depending on the node, this menu displays the following entries:

**Note:** The right-click context menu options are not available for the elements that are part of read-only code sections.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cut</strong></td>
<td>Cut the selected node.</td>
</tr>
<tr>
<td><strong>Note:</strong> This option is not supported for Edit Freeform Query.</td>
<td></td>
</tr>
<tr>
<td><strong>Copy</strong></td>
<td>Copy the selected node.</td>
</tr>
<tr>
<td><strong>Paste</strong></td>
<td>Paste the selected node.</td>
</tr>
<tr>
<td><strong>Note:</strong> This option is not supported for Edit Freeform Query.</td>
<td></td>
</tr>
<tr>
<td><strong>Duplicate</strong></td>
<td>Create a copy of the selected widget.</td>
</tr>
<tr>
<td><strong>Note:</strong> This option is available only for widgets present on the design window open.</td>
<td></td>
</tr>
<tr>
<td><strong>Delete</strong></td>
<td>Delete a selected node.</td>
</tr>
<tr>
<td><strong>Note:</strong> This option is not supported for Edit Freeform Query.</td>
<td></td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Enable DB-Required / Disable</strong></td>
<td>Toggle switch to add or remove DB-Required block generated for a function and procedure. See the Specifying DB-Required code blocks on page 675.</td>
</tr>
<tr>
<td><strong>DB-Required</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Move-to-Top</strong></td>
<td>Makes the selected basic object or SmartObject instance overlay other basic objects or SmartObject instances of its type.</td>
</tr>
<tr>
<td><strong>Move-to-Bottom</strong></td>
<td>Makes the selected basic object or SmartObject instance underlay other basic objects or SmartObject instances of its type.</td>
</tr>
<tr>
<td><strong>Properties</strong></td>
<td>Opens the Property Sheet dialog for the selected widget.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>This option is available only with widgets present on the design window open.</td>
</tr>
<tr>
<td><strong>Add Trigger</strong></td>
<td>Opens the Add Trigger dialog on page 757 that allows you to add a new trigger for the selected widget. The Add Trigger menu option is disabled for the SmartObject instances nodes in the Outline view.</td>
</tr>
<tr>
<td><strong>Define Service Interface</strong></td>
<td>Opens the Define Service Interface wizard that allows you to publish the selected procedures and functions as a service interface.</td>
</tr>
<tr>
<td><strong>Edit Freeform Query</strong></td>
<td>Opens the Edit Freeform Query dialog box that allows you to edit the code of the Query.</td>
</tr>
<tr>
<td><strong>Caution:</strong></td>
<td>If you are not connected to the database, content assist and color coding will not work as expected.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>This option is available with the view source mode only.</td>
</tr>
</tbody>
</table>

**See also**
- Using the Outline view on page 657
- Outline view toolbar buttons on page 813
ABL Cue Cards view

The ABL Cue Card view displays help information for a specific SmartObject type. When you create a new SmartObject master, the Cue Card associated with the SmartObject type is also opened. The Cue Card provides a basic definition of the SmartObject type, along with information about how to create and use the object. This view lists details of all the active Cue Cards in your current AppBuilder session.

Note: The ABL Cue Cards is the default view and is available only with the OpenEdge AppBuilder perspective.

To retrieve the ABL Cue Cards view that you close, choose Help > ABL Cue Cards from the main menu.

See also
Outline view on page 813
ABL Messages view on page 816

ABL Messages view

You can use the ABL Messages view to look up information about ABL error messages.

Note: The ABL Messages view is the default view in OpenEdge AppBuilder perspective.

This view includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter message number</td>
<td>Enter an error message number.</td>
</tr>
<tr>
<td>Go</td>
<td>Click to locate and display the message text for the error message number entered in the Enter message number field.</td>
</tr>
<tr>
<td>Search result</td>
<td>Displays the synopsis, category, and description for the error message number.</td>
</tr>
</tbody>
</table>

For more information, see Viewing ABL messages on page 673

Other OpenEdge Tools

From Progress Developer Studio for OpenEdge, you can also access the Procedure Editor, Data Dictionary, and many other OpenEdge tools that are not Eclipse applications.

Most of these tools do not run within Progress Developer Studio for OpenEdge, however. When started, they run in a window outside the Eclipse framework.

The exception is OpenEdge Explorer, which runs in an embedded browser.

Online help for these tools is available from a tool's user interface and not from Progress Developer Studio's help menu.
Audit Policy Maintenance Tool (Windows only)

The Audit Policy Maintenance Tool is a utility that enables you to create and maintain audit policies for database events, application events, and internal (to OpenEdge) system events.

To start the Audit Policy Maintenance Tool, select OpenEdge > Admin > Audit Policy Maintenance from the menu bar in any Progress Developer Studio for OpenEdge perspective except the Progress DB Navigator and the OpenEdge Debugger.

When started, the Audit Policy Maintenance tool runs in a window outside of the Progress Developer Studio for OpenEdge Eclipse framework. Online help is available from the tool's menu bar.

For more information, see Other OpenEdge Tools on page 816

Temp-DB Maintenance Tool

The Temp-DB Maintenance tool opens the Connect Database dialog. This dialog prompts you for the connection parameters and provides a scrolling area for additional connection parameters.

To start the Temp-DB Maintenance tool, select OpenEdge > Tool > Temp-DB > Maintenance from the main menu in your OpenEdge AppBuilder perspective.

When started, the Temp-DB Maintenance tool runs in a window outside the Progress Developer Studio Eclipse framework. Online help is available from the tool's menu bar.

For more information, see Other OpenEdge Tools on page 816

Database Administration Tool

The Database Administration Tool is tool used for dumping and loading data and definitions, exchanging definition information from non-OpenEdge data sources, defining application security and permissions, and importing and exporting data from a variety of sources.

To start the Database Administration Tool, select OpenEdge > Admin > Database Administration from the menu bar in any OpenEdge perspective except DB Navigator and Debug.

When started, the Database Administration tool runs in a window outside the Progress Developer Studio Eclipse framework. Online help is available from the tool's menu bar.

For more information, see Other OpenEdge Tools on page 816

Data Dictionary tool

The Data Dictionary is a tool that lets you create and modify database schema information including table, field, sequence, index, and trigger definitions.

Note: The Data Dictionary has functionality that is similar to DB Navigator. Consider using DB Navigator since it is integrated with Progress Developer Studio and is therefore more accessible to your project.

To start the Data Dictionary, select OpenEdge > Admin > Data Dictionary from the menu bar of any OpenEdge perspective except DB Navigator and Debug.

When started, the Data Dictionary runs in a window outside the Progress Developer Studio Eclipse framework. Online help is available from the tool's menu bar.

For more information, see Other OpenEdge Tools on page 816
OpenEdge Explorer Tool

OpenEdge Explorer is a tool for managing OpenEdge servers and other products.

To start OpenEdge Explorer, select OpenEdge > Admin > OpenEdge Explorer from the menu bar of any Progress Developer Studio for OpenEdge perspective except Progress DB Navigator and OpenEdge Debugger.

When started, OpenEdge Explorer prompts for an administrator user name and password. The default is admin for both. After login, it runs in a browser embedded in the active perspective. For online help, click one of the help buttons in OpenEdge Explorer.

For more information, see Other OpenEdge Tools on page 816

Procedure Editor tool

The Procedure Editor is an OpenEdge ABL code editor.

Note: When you run the Procedure Editor from Progress Developer Studio for OpenEdge, advanced editing features (color coding, syntax expansion, and so on) are not available. These advanced features are available in the OpenEdge Studio version of the Procedure Editor. For a full-featured, integrated editor, use the ABL Editor.

Use one of the following methods to open the Procedure Editor:

• Click the Procedure Editor icon in the toolbar.
• Select OpenEdge > Tools > Procedure Editor from the menu bar.

If you have a file selected in the Project Explorer view, the selected file opens in the Procedure Editor.

When started, the Procedure Editor runs in a window outside the Progress Developer Studio Eclipse framework. Online help is available from the tool's menu bar.

For more information, see Other OpenEdge Tools on page 816

Results Tool

The Results tool is used to create and modify queries. A query is a set of instructions that tells Results to find the contents of one or more tables in your database. Creating a query consists of selecting the following:

• The view you want to use to process the information in tables that comprise the query.
• The query.
• Fields you want to display.

To start the Results tool, select OpenEdge > Tool > Results from the main menu in your OpenEdge AppBuilder perspective.

Note: You are prompted to connect to a database when you start Results tool, if the database connection is not configured with your current project. When you click OK, the Connect Database dialog appears.

When started, the Results tool runs in a window outside the Progress Developer Studio Eclipse framework. Online help is available from the tool's menu bar.

For more information, see Other OpenEdge Tools on page 816
**Translation Manager Tool**

The Translation Manager tool is used to select the strings, data formats, and the interface of an ABL application to translate into one or more languages. You can also use the Translation Management System to customize the interface of an application for different customer sites. This tool works in partnership with the OpenEdge Visual Translator tool.

The OpenEdge Translation Management System consists of two tools:

- **OpenEdge Translation Manager** - lets you (the project manager) select procedures to translate, filter and extract text phrases, and create or add glossaries.
- **OpenEdge Visual Translator** - lets a translator translate the text phrases in context or from a list.

---

**Note:** The OpenEdge Translation Manager and the OpenEdge Visual Translator run in single-user mode only.

To start the Translation Manager tool, select **OpenEdge > Tool > Translation Manager** from the main menu in your OpenEdge AppBuilder perspective.

When started, the Translation Manager tool runs in a window outside the Progress Developer Studio Eclipse framework. Online help is available from the tool's menu bar.

For more information, see Other OpenEdge Tools on page 816

---

**Visual Translator Tool**

The Visual Translator tool is used to translate the text phrases, data formats, and the interface of an ABL application into one or more languages. This tool works in partnership with the OpenEdge Translation Manager tool.

To start the Visual Translator tool, select **OpenEdge > Tool > Visual Translator** from the main menu in your OpenEdge AppBuilder perspective.

When started, the Visual Translator tool runs in a window outside the Progress Developer Studio Eclipse framework. Online help is available from the tool's menu bar.

For more information, see Other OpenEdge Tools on page 816
Introducing the Meta Catalog

The Progress Developer Studio for OpenEdge's Meta Catalog is an index that enables you to find where elements are used in your application. You can find where a temp-table is defined and where it is used in your application. You can find all the procedures and functions in your application. You can find where those procedures and functions are called. You can also add your own annotations to the code and have them included in the index. You can use this index to simplify analyzing the impact of proposed changes and carrying out those changes.

You can configure different catalogs for different tasks. You might want to have the data for each project in a different catalog. Alternately, you might want a catalog that stores data on function and procedure calls for all your projects together. You can also configure a master catalog linked to your software code management (SCM) system to provide a complete view of all your applications.

The Meta Catalog is a design-time tool. A catalog never stores any data that cannot be extracted from the source code by the content builders. Any time the catalog has stale data in it, you can eliminate the stale data by replacing the catalog. For example, if you remove files from a project and no longer want data from those files in the catalog, just rebuild the catalog using the catalog's definition from the Meta Catalog preferences.

The Meta Catalog uses content builders to extract the data from source files. The Progress Developer Studio for OpenEdge provides several predefined content builders.

There are two tools for searching through a catalog's data. The Meta Catalog Explorer provides a treeview representation of the data. The Meta Catalog Search allows you to create and save queries on the data. You can open files for editing from either the treeview or the Meta Catalog Search results view.

Related Concepts

For details, see the following topics:

• Concepts
• Tasks
Chapter 15: Introducing the Meta Catalog

- Reference

Concepts

**ABL Annotations**

ABL annotations enable you to include meta data in your source code. The compiler treats annotations as comments, so they do not affect how your code runs. The ABL Annotation content builder extracts annotations from your source code and loads them into the Meta Catalog.

You can assign almost any name to an annotation.

You can insert annotations manually in your source code. You can also use the Annotation Generator to insert the same annotation into multiple files at once. The Annotation Generator always inserts the annotations either at the top of the file or, for structured procedures, in the definitions section at the start of the file.

**See also**
- ABL Annotation content builder on page 831
- ABL annotation syntax on page 832
- Annotation Generator on page 839

**Access to catalog data**

The Meta Catalog provides two tools for examining the data in a catalog. The Meta Catalog Explorer view provides a treeview approach to searching a catalog. The Meta Catalog Search allows you to write queries on the catalog content’s and displays the results in a search view.

**See also**
- Introducing the Meta Catalog on page 821
- Accessing the Meta Catalog Explorer on page 824
- Searching the Meta Catalog on page 829
- Meta Catalog Explorer view on page 842
- Meta Catalog Search view on page 847

**Content builders**

Content builders extract data from the source code and store it in the Meta Catalog. Each content builder looks for specific code segments, called content types, and retrieves matching data from the source code.

Progress Developer Studio for OpenEdge provides several predefined content builders. A content builder extracts data for one or more content types for storage in the Meta Catalog.

The Meta Catalog design allows for additional content builders that extract more data from the source code. For example, a content builder could be designed to extract index usage information. The extensibility of the content builder mechanism provides the possibility that it could be extended to extract data from non-ABL application code if necessary.

Predefined content builders installed with Progress Developer Studio for OpenEdge
Progress Developer Studio for OpenEdge provides the following predefined content builders:

<table>
<thead>
<tr>
<th>Predefined Content Builders</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABL Annotation</td>
<td>Extracts annotations that you either manually add to the source code or included through the Annotation Generator.</td>
</tr>
<tr>
<td>ABL Application Schema</td>
<td>Extracts data on temp-tables and ProDataSets.</td>
</tr>
<tr>
<td>ABL Reference</td>
<td>Extracts data on procedures, functions, methods, and where they are called.</td>
</tr>
</tbody>
</table>

See also
Introducing the Meta Catalog on page 821
Adding custom file extensions to content builders on page 824
ABL Annotation content builder on page 831
ABL annotation syntax on page 832
ABL Application Schema content builder on page 832
ABL Reference content builder on page 836

Local and master catalogs

In many development environments, you do not have the complete application source code on your local machine. Environments that include a source code management (SCM) system are like this. However, to gain the most advantage from a catalog, it should incorporate information on your entire application.

The OpenEdge Meta Catalog supports linking catalogs to your workspace in two modes:

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local</td>
<td>A local catalog is specific to your workspace. It includes information only from the projects you specify in that workspace. You can update the contents of a local catalog at any time. You can create more than one catalog in a workspace, using each to store information for a particular project, content builder, or other criteria.</td>
</tr>
<tr>
<td>Master</td>
<td>A master catalog exists outside your workspace. You have read-only access in this mode. Changes to your local files do not affect the contents of a master catalog. A group of developers can all use the same master catalog. A good use for a master catalog is to index all resources in a SCM system. Each developer can perform impact analysis across an entire application without needing all the source-code locally. The catalog could update its contents each time a changed file is checked into the SCM or at scheduled times. The SCM administrator builds a catalog database and then sets up scripts to update it appropriately using the Meta Catalog command-line utility.</td>
</tr>
</tbody>
</table>

See also
Introducing the Meta Catalog on page 821
Building a master catalog on page 825
Tasks

Accessing the Meta Catalog Explorer

Accessing the Meta Catalog Explorer

1. Choose **Window > Show View > Other...** to display the **Show View** dialog box.
2. Choose **Progress OpenEdge Meta Catalog > Meta Catalog Explorer**.
3. Choose **OK**.

The **Meta Catalog Explorer** appears, displaying all the catalogs defined for your workspace.

**Note:** If you want access to a catalog that is defined in a different workspace, you must define it as a master catalog in your workspace.

See also

- Access to catalog data on page 822
- Using the Meta Catalog Explorer on page 822
- Meta Catalog Explorer view on page 842

Adding custom file extensions to content builders

By default, the content builders run against files with the standard OpenEdge extensions (*.cls, *.i, *.p, and *.w). You can configure them to run against other files with custom extensions. For example, if you use .t as the file extension for your ABL trigger source code, you should add the .t extension to the list of extensions to ensure that the content builder runs against your trigger source code.

1. Choose **Window > Preferences**. The **Preferences** dialog appears.
2. Select **Progress OpenEdge > Meta Catalog > Content Builders**. The **Content Builders** page appears.
3. Select a content builder in the browse.
4. Choose **Edit**. The **Edit Content Builder** dialog appears.
5. Add your custom extension to the list in the **Extensions field**.
6. Choose **OK**.
7. Choose **Apply** to save your changes to the internal Meta Catalog definition file.

**Note:** The content builder definitions are also stored when you export catalog definitions to a file. Importing an existing definition file will overwrite any changes you made to the content builder definitions. If you do not want that to happen, you should export the catalog definitions again to overwrite the old file.
Adding new files to a catalog

You can add new source files to a catalog without rebuilding the entire catalog. You can also use this technique to force the content builders to rescan a file and update the catalog with any new information.

1. Select a file in the **Project Explorer** view.
2. Right-click and choose **Meta Catalog > Add Selected Resource** from the pop-up menu.

**Note:** The data from the file is added to all local catalogs associated with the workspace.

See also

- Changing catalog definitions on page 826
- Rebuilding a local catalog on page 828

Building a master catalog

The **OECATALOG** command-line utility’s **-build** option does not create the physical database. That option only creates the tables and populates them with data. To create a new master catalog, you MUST run the **New Catalog** wizard on the physical database before using the **OECATALOG** command-line utility.

Building a master catalog on a server is a multi-stage process:

1. Create an empty database and database server on the server.
   a) Start a **PROENV** session on the machine where you want the master catalog.
   b) Create a directory to hold the master catalog.
   c) Use the **PRODB** utility to create an **Empty database** in that directory.
   d) Create a database server with the appropriate connection parameters for the master catalog using OpenEdge Management or OpenEdge Explorer. Remember that the database server has to support SQL.
   e) Start the database server.

2. Run the New Catalog wizard
   a) Start a Progress Developer Studio for OpenEdge session on your local machine in a workspace that has no catalogs defined.
   b) Choose **File > New > Other**. The **New** dialog box appears.
   c) Choose **OpenEdge > Meta Catalog > Meta Catalog** and click **Next**. The **Create Meta Catalog** wizard appears.
   d) Enter the empty database's name for the Meta Catalog name and click **Next**.
   e) Click **Next** on the this page. You have already created the database.
f) On the Meta Catalog Connection page, enter the connection information for the empty database and click Next.

g) Continue creating the catalog definition as usual.

h) When you have set all the options correctly for your master catalog, deselect the Load content from selected projects toggle and click Finish.

Your master catalog now has the Meta Catalog tables.

3. Transfer the catalog definition from your local machine to the server.
   Before you can use the OECATALOG utility on your master catalog, you need a catalog definition file for it.
   a) Choose Windows > Preferences. The Preferences dialog box appears.
   b) Choose OpenEdge > Meta Catalog. The Meta Catalog browser appears.
   c) Click Export All.
   d) Save the catalog definition file with a unique name.
   e) Copy the catalog definition file to the master catalog's directory on the server.

   You now have a master catalog database and a catalog definition file for it.

4. Run the OECATALOG command-line utility to populate the master catalog using the catalog definition file.
   For example: oecatalog -build -catalog masterCatalog.xml -root c:\wrk

   Note: The -root setting should match where the resources are stored on the master catalog's machine, not where your local copies are.

You can now use the OECATALOG utility to populate and rebuild the master catalog as needed. The master catalog does not contain any data yet, because the content builders have not run against the source files yet.

See also
Local and master catalogs on page 823
Creating a local catalog on page 827
Linking to a master catalog on page 828
OECATALOG utility on page 848

Changing catalog definitions

After creating a catalog, you can change its definition at any time through the Meta Catalog Preferences.

1. Choose Window > Preferences. The Preferences dialog appears.
2. Select Progress OpenEdge > Meta Catalog > Catalog Databases. The Catalog Databases page appears.
3. Select a catalog in the Catalog Name combo-box. The tab folder displays the catalog's definition.
4. Make your changes to the definition.
5. Choose Apply to store the changes to the internal Meta Catalog definition file. You might need to restart your session for some changes to take effect.
**Note:** If you have exported the previous catalog definition to a file, you can import that file to revert to the previous definition. If you do not want that to happen, you should export the catalog definitions again to overwrite the old file.

**See also**
- Adding custom file extensions to content builders on page 824
- Rebuilding a local catalog on page 828
- Meta Catalog Preferences on page 843

**Creating a local catalog**

1. Choose **File > New > Other** to display the **New** dialog box.
2. Choose **OpenEdge > Meta Catalog > Meta Catalog**.
3. Choose **Next** to launch the **Create Meta Catalog** wizard.
4. Fill in a Meta Catalog name and choose **Next**.
   - All catalog names must be unique and are case-sensitive. If you want to use a master catalog, you cannot have a local catalog with the same name.
5. Select the **Create new database** toggle box and fill in the information for the new database.
6. If you want to create a catalog with the default settings, choose **Finish**. Otherwise, choose **Next** to step through the remaining wizard pages and customize the catalog.

You can change the definition of a catalog later through the **Preferences** window.

**See also**
- Local and master catalogs on page 823
- Adding new files to a catalog on page 825
- Building a master catalog on page 825
- Changing catalog definitions on page 826
- Rebuilding a local catalog on page 828
- Meta Catalog preferences on page 843

**Creating templates for the Annotation Generator**

Creating templates for the Annotation Generator

1. Choose **Window > Preferences**.
2. Choose **Progress OpenEdge > Editor > Annotations**.
3. Choose **Add**. The **New Annotation** dialog appears.
4. Type an annotation name for the new template.
5. Type an annotation for the Annotation Value. You can create multi-line templates.
   - The dialog does not check the syntax of the template.
6. Choose **OK** to save the template.
Linking to a master catalog

1. Choose File > New > Other to display the New dialog box.
2. Choose OpenEdge > Meta Catalog > Meta Catalog to link to Master.
3. Choose Next to launch the Add Meta Catalog wizard.
4. Fill in a Meta Catalog name and choose Next. The Meta Catalog Connection page appears.
   All catalog names must be unique and are case sensitive. You cannot have a local catalog with the same
   name as the master catalog.
5. Fill in the connection information for the master catalog.

Note: Your workspace cannot automatically start a master catalog.

Rebuilding a local catalog

If you delete files from a project, the catalog might contain obsolete data from those files. If you move code
from one file to another file, the catalog might still have records pointing to the old location. Catalogs are
designed to be disposable. Whenever necessary, you can rebuild a catalog from its definition.

1. Choose Window > Preferences to open the Preferences window.
2. Choose Progress OpenEdge > Meta Catalog in the treeview. The Meta Catalog page appears.
3. Select a catalog from the browse.
4. Choose Rebuild.

The process deletes the existing database and rebuilds it from the definition.
Searching the Meta Catalog

Searching the Meta Catalog

1. Choose **Search** to open the Search view.
2. Select the **Meta Catalog Search** tab.
3. Select the projects whose contents you want to search from the Projects to search list.
4. Select the catalogs whose contents you want to search from the Catalogs to search list.
5. Define your search in the Search criteria browse. You can define the search in the following ways:
   a) Select a previously created query in the Query criteria combo-box. You can alter the saved criteria before running the search.
   b) Create a new query by selecting entries from the combo-boxes that make up the Query criteria browse.
6. Choose **Search**. The search results appear in a treeview.
   If there is a name in the **Query criteria** combo-box, a prompt appears asking if you want to save the current criteria under that name.
7. Double-click an item in the search results. The file opens in the Editor view positioned at the line containing the item.

If the same file's contents have been loaded into two catalogs, the search results show duplicate entries for that data.

See also

- Access to catalog data on page 822
- Accessing the Meta Catalog Explorer on page 824
- Using the Meta Catalog Explorer on page 830
- Meta Catalog Search view on page 847

Setting Meta Catalog preferences

1. Choose **Window** > **Preferences** to open the **Preferences** window.
2. Choose **Progress OpenEdge > Meta Catalog** in the treeview.
3. Expand the **Meta Catalog** node to access all the preference pages.

The Meta Catalog preferences appear on the following pages:

- **Catalog List**- A browse listing all the catalogs in your workspace.
- **Catalog Databases**- A tab folder for viewing and editing catalog definitions.
• **Content Builders** - A browse of all the content builders registered in your workspace.

**See also**
- Introducing the Meta Catalog on page 821
- Changing catalog definitions on page 826
- Adding custom file extensions to content builders on page 824
- Meta Catalog preferences on page 843

## Using the Meta Catalog Explorer

Using the Meta Catalog Explorer

1. Open the **Meta Catalog Explorer** view. The treeview displays a node for each catalog defined in the workspace.

2. Select a **CATALOG** node and expand it. The treeview displays a node for each root path in the catalog.

   **Note**: The first line in the browse always shows the node's type. These are the values you use when constructing queries in the Meta Catalog Search. The rest of the browse displays the other database fields that define the node in the catalog.

3. Expand a **ROOT PATH** node. The treeview displays a node for each project stored on that root path.

4. Expand a **PROJECT** node. The treeview displays nodes for project relative paths and individual resources.

   **Note**: If you select the **Create complete folder structure** option while importing files into a project, the structure is stored as a **PROJECT RELATIVE PATH** in the catalog. All the resources in a folder structure display under the **PROJECT RELATIVE PATH**. If there is no relative path for a particular resource, the resource displays directly under the **PROJECT** node.

5. Expand a **RESOURCE** node. The treeview displays nodes for each content type that a content builder extracted from the file.

6. Expand a **CONTENT TYPE** node. The treeview displays various **ENTRY TYPE** nodes, depending on which content builders extracted the data and what was in the file. For more information on the finer levels of the tree view, see the description of each content builder.

   **Note**: The treeview is recursive when displaying Parent Entries and Child Entries nodes. If you expand a Child Entries node, you see a Parent Entries node. If you expand a Parent Entries node, you see a Child Entries node. These nodes will loop infinitely.

**See also**
- Access to catalog data on page 822
- Accessing the Meta Catalog Explorer on page 824
- Searching the Meta Catalog on page 829
- ABL Annotation content builder on page 831
- ABL Application Schema content builder on page 832
- ABL Reference content builder on page 836
- Meta Catalog Explorer view on page 842
ABL Annotation content builder

The ABL Annotation content builder extracts ABL annotations from source files. The content builder searches for the "at" symbol (@) at the beginning of a line. When it finds the symbol, it parses the line as an annotation. The content builder first ensures that the catalog database contains the necessary records to specify the source file (catalog, root path, project, project relative path, and resource). It then records the following data in the catalog database:

<table>
<thead>
<tr>
<th>Data</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content type</td>
<td>ANNOTATION</td>
</tr>
<tr>
<td>Entry type</td>
<td>ANNOTATION</td>
</tr>
<tr>
<td>Entry</td>
<td>The annotation name</td>
</tr>
<tr>
<td>Attributes</td>
<td>ATTRIBUTES</td>
</tr>
<tr>
<td>Attribute</td>
<td>The attribute name</td>
</tr>
<tr>
<td>Attribute value</td>
<td>The attribute's value</td>
</tr>
</tbody>
</table>

The content builder assigns IDs to each unique Entry and Attribute. It also stores the location of these elements in the file, either as a line number or as characters from the file's beginning.

For example, the following annotation was created from one of the templates in the Annotation Generator:

`@program (name = "inventory.p", module = "warehouse")`

When the content builder analyzes this statement, it adds the following information to the database:

<table>
<thead>
<tr>
<th>Data</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content type</td>
<td>Annotation</td>
</tr>
<tr>
<td>Entry type</td>
<td>Annotation</td>
</tr>
<tr>
<td>Entry</td>
<td>program</td>
</tr>
<tr>
<td>Attributes</td>
<td>Attributes</td>
</tr>
<tr>
<td>Attribute</td>
<td>name</td>
</tr>
<tr>
<td>Attribute value</td>
<td>inventory.p</td>
</tr>
<tr>
<td>Attribute</td>
<td>module</td>
</tr>
<tr>
<td>Attribute value</td>
<td>warehouse</td>
</tr>
</tbody>
</table>
In the Meta Catalog Explorer, you would see the following structure:

![Meta Catalog Explorer Screenshot]

See also

- Content builders on page 822
- ABL Annotations on page 822
- Adding custom file extensions to content builders on page 824
- Creating templates for the Annotation Generator on page 827
- ABL Annotation Syntax on page 832
- Annotation Generator on page 839

ABL annotation syntax

ABL annotations are always on a separate line from other code. The compiler treats any line that starts with the "at" symbol ( @ ) as a ABL annotation.

Note: Annotation syntax must begin with a non-numeric.

The ABL Annotation syntax is as follows: @annotation[(attribute = "value", attribute = "value")].

- **annotation**: The annotation's name can be any character string that you choose.
- **attribute**: The attribute's name can be any character string that you choose. Attribute/value pairs are optional.
- **value**: The value can be any character string that you choose. Attribute/value pairs are optional.

See also

- Content builders on page 822
- ABL annotations on page 822
- Creating templates for the Annotation Generator on page 827
- ABL Annotation content builder on page 831
- Annotation Generator on page 839

ABL Application Schema content builder

ABL Application Schema content builder

The ABL Application Schema content builder extracts data on temp-tables and ProDataSets from source files. The content builder first ensures that the catalog database contains the necessary records to specify the source file (catalog, root path, project, project relative path, and resource). Then it analyzes the code to extract the specific information.
The content builder creates parent and child records to describe the full schemas:

<table>
<thead>
<tr>
<th>Entry type</th>
<th>Child entry types</th>
<th>Child entry types</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEMP-TABLE</td>
<td>FIELD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>INDEX</td>
<td>INDEX-FIELD</td>
</tr>
<tr>
<td>DATASET</td>
<td>BUFFER</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DATA-RELATION</td>
<td>RELATION-FIELD</td>
</tr>
</tbody>
</table>

Any entry that has parent or child entries displays them as subnodes in the Meta Catalog Explorer view.

The content builder records the following data in the catalog:

<table>
<thead>
<tr>
<th>Content type</th>
<th>Entry type</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPSCHEMA</td>
<td>TEMP-TABLE</td>
<td>The temp-table's name.</td>
</tr>
<tr>
<td></td>
<td>FIELD</td>
<td>The name of a temp-table field.</td>
</tr>
<tr>
<td></td>
<td>INDEX</td>
<td>The name of a temp-table index.</td>
</tr>
<tr>
<td></td>
<td>INDEX-FIELD</td>
<td>The name of a field in a temp-table index.</td>
</tr>
<tr>
<td></td>
<td>DATASET</td>
<td>The ProDataSet's name.</td>
</tr>
<tr>
<td></td>
<td>BUFFER</td>
<td>The name of a ProDataSet buffer.</td>
</tr>
<tr>
<td></td>
<td>DATA-RELATION</td>
<td>The name of a ProDataSet data relationship.</td>
</tr>
<tr>
<td></td>
<td>RELATION-FIELD</td>
<td>The name of a field in a ProDataSet data relationship.</td>
</tr>
</tbody>
</table>

Depending on the entry type, the content builder also extracts the following attribute information for each entry:
<table>
<thead>
<tr>
<th>Entry type</th>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEMP-TABLE</td>
<td>NO-UNDO</td>
<td>TRUE or FALSE.</td>
</tr>
<tr>
<td></td>
<td>GLOBAL</td>
<td>TRUE if a global temp table.</td>
</tr>
<tr>
<td></td>
<td>SHARED</td>
<td>TRUE if a shared temp table.</td>
</tr>
<tr>
<td></td>
<td>LIKE</td>
<td>TRUE if defined with the LIKE option.</td>
</tr>
<tr>
<td>FIELD</td>
<td>DATA-TYPE</td>
<td>The data type.</td>
</tr>
<tr>
<td></td>
<td>LIKE</td>
<td>TRUE if defined with the LIKE option.</td>
</tr>
<tr>
<td></td>
<td>EXTENT</td>
<td>TRUE if extents exist.</td>
</tr>
<tr>
<td></td>
<td>FORMAT</td>
<td>The format, if any.</td>
</tr>
<tr>
<td></td>
<td>LABEL</td>
<td>The label, if any.</td>
</tr>
<tr>
<td></td>
<td>INITIAL</td>
<td>The initial value, if any.</td>
</tr>
<tr>
<td>INDEX</td>
<td>PRIMARY</td>
<td>TRUE if a primary index.</td>
</tr>
<tr>
<td></td>
<td>UNIQUE</td>
<td>TRUE if a unique index</td>
</tr>
<tr>
<td></td>
<td>WORD-INDEX</td>
<td>TRUE if a word index.</td>
</tr>
<tr>
<td>INDEX-FIELD</td>
<td>SORT</td>
<td>ASCENDING or DESCENDING.</td>
</tr>
<tr>
<td>DATASET</td>
<td>SHARED</td>
<td>TRUE if a shared ProDataSet.</td>
</tr>
<tr>
<td>DATA-RELATION</td>
<td>REPOSITION</td>
<td>TRUE if applicable.</td>
</tr>
</tbody>
</table>

The content builder assigns IDs to each unique Entry and Attribute. It stores the location of these elements in the file, either as a line number or as characters from the file's beginning. The content builder also stores pointers to any parent or child entries.

For example, take the following temp-table definition:
• DEFINE SHARED TEMP-TABLE ttOrd
• FIELD OrderNum AS DECIMAL
• FIELD CustName AS CHARACTER LABEL "Name"
• FIELD OrderDate LIKE Order.OrderDate
• INDEX OrderNum IS PRIMARY UNIQUE OrderNum
• INDEX CustOrder IS UNIQUE CustName OrderNum
• INDEX OrderDate OrderDate.

When the content builder analyzes this statement, it adds the following information to the database:

<table>
<thead>
<tr>
<th>Content Type</th>
<th>Entry type</th>
<th>Entry</th>
<th>Attributes</th>
<th>Attribute values</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPSCHEMA</td>
<td>TEMP-TABLE</td>
<td>ttOrd</td>
<td>SHARED</td>
<td>TRUE</td>
</tr>
<tr>
<td>FIELD</td>
<td>OrderNum</td>
<td>DATA-TYPE</td>
<td>DECIMAL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CustName</td>
<td>DATA-TYPE</td>
<td>CHARACTER</td>
<td>&quot;Name&quot;</td>
</tr>
<tr>
<td></td>
<td>OrderDate</td>
<td>LIKE</td>
<td>TRUE</td>
<td></td>
</tr>
<tr>
<td>INDEX</td>
<td>OrderNum</td>
<td>PRIMARY</td>
<td>TRUE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UNIQUE</td>
<td>TRUE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CustOrder</td>
<td>UNIQUE</td>
<td>TRUE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OrderDate</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INDEX-FIELD</td>
<td>CustName</td>
<td>SORT</td>
<td>ASCENDING</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OrderDate</td>
<td>SORT</td>
<td>ASCENDING</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OrderNum</td>
<td>SORT</td>
<td>ASCENDING</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OrderNum</td>
<td>SORT</td>
<td>ASCENDING</td>
<td></td>
</tr>
</tbody>
</table>

There are two OrderNum entries because two indexes use it.

As mentioned previously, the content builder also stores the data on any parent or child relationships for each entry.

When displayed in the Meta Catalog Explorer, you would see the following structure:
ABL Reference content builder

The ABL Reference content builder extracts data on procedures, functions, and methods. The content builder first ensures that the catalog database contains the necessary records to specify the source file (catalog, root path, project, project relative path, and resource). The content builder analyzes the code to find where procedures, functions, and methods are defined, what their parameters are, and where they are used.

Each parameter is stored as a child record for the appropriate procedure, function, or method. Any entry that has parent or child entries displays them as subnodes in the Meta Catalog Explorer view.

The content builder records the following data in the catalog:

See also

- Content builders on page 822
- Adding custom file extensions to content builders on page 824
<table>
<thead>
<tr>
<th>Content type</th>
<th>Entry type</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABLREFERENCE</td>
<td>PROCEDURE</td>
<td>The procedure name.</td>
</tr>
<tr>
<td></td>
<td>FUNCTION</td>
<td>The function name.</td>
</tr>
<tr>
<td></td>
<td>METHOD</td>
<td>The method name.</td>
</tr>
<tr>
<td></td>
<td>PARAMETER</td>
<td>The parameter name.</td>
</tr>
<tr>
<td></td>
<td>PROCEDURE-REF</td>
<td>The name of the referenced procedure.</td>
</tr>
<tr>
<td></td>
<td>FUNCTION-REF</td>
<td>The name of the referenced function.</td>
</tr>
<tr>
<td></td>
<td>METHOD-REF</td>
<td>The name of the referenced method.</td>
</tr>
</tbody>
</table>

Depending on the entry type, the content builder also extracts the following attribute information for each entry:

<table>
<thead>
<tr>
<th>Entry type</th>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROCEDURE</td>
<td>ACCESS-MODIFIER</td>
<td>PRIVATE</td>
</tr>
<tr>
<td>PARAMETER</td>
<td>PARAMETER-TYPE</td>
<td>INPUT, OUTPUT, or INPUT/OUTPUT.</td>
</tr>
<tr>
<td></td>
<td>DATA-TYPE</td>
<td>The parameter's data type.</td>
</tr>
<tr>
<td></td>
<td>LIKE</td>
<td>TRUE if defined with the LIKE option.</td>
</tr>
<tr>
<td></td>
<td>EXTENT</td>
<td>The extents, if any.</td>
</tr>
<tr>
<td></td>
<td>NO-UNDO</td>
<td>TRUE if set.</td>
</tr>
<tr>
<td>FUNCTION</td>
<td>RETURN-DATA-TYPE</td>
<td>The data type returned by the function.</td>
</tr>
<tr>
<td></td>
<td>ACCESS-MODIFIER</td>
<td>PRIVATE</td>
</tr>
<tr>
<td>METHOD</td>
<td>RETURN-DATA-TYPE</td>
<td>The data type returned by the function.</td>
</tr>
<tr>
<td></td>
<td>ACCESS-MODIFIER</td>
<td>PUBLIC, PROTECTED, or PRIVATE</td>
</tr>
</tbody>
</table>
The content builder assigns IDs to each unique Entry and Attribute. It stores the location of these elements in the file, either as a line number or as characters from the file's beginning. The content builder also stores pointers to any parent or child entries.

For example, take the following procedure:

```
PROCEDURE activateSession:
  DEFINE INPUT PARAMETER pcSessType AS CHARACTER NO-UNDO.
  RUN storeAppServerInfo IN TARGET-PROCEDURE
    (":U).
  DYNAMIC-FUNCTION("setSessionParam":U IN TARGET-PROCEDURE,
    "client_SessionType":U,
    pcSessType).
END PROCEDURE.
```

When the content builder analyzes this statement, it adds the following information to the database:

<table>
<thead>
<tr>
<th>Content Type</th>
<th>Entry type</th>
<th>Entry</th>
<th>Attributes</th>
<th>Attribute values</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABLREFERENCE</td>
<td>PROCEDURE</td>
<td>activateSession</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PARAMETER</td>
<td>pcSessType</td>
<td>PARAMETER-TYPE</td>
<td>INPUT</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DATA-TYPE</td>
<td>CHARACTER</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NO-UNDO</td>
<td>TRUE</td>
</tr>
<tr>
<td></td>
<td>PROCEDURE-REF</td>
<td>storeAppServerInfo</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FUNCTION-REF</td>
<td>setSessionParam</td>
<td>DYNAMIC</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

As mentioned previously, the content builder also stores the data on any parent or child relationships for each entry.

When displayed in the Meta Catalog Explorer, you would see the following structure:
Annotation Generator

The Annotation Generator enables you to insert basic annotations into multiple files. The Annotation Generator always inserts the annotations either at the first line of the file or, for AppBuilder structured files, in the definitions section at the top of the file. This placement means that it is appropriate for adding annotations that apply to the entire file, rather than a particular piece of the file. For example, you might use the tool to insert annotations for who wrote the code, when it was created, or the sort of deployments in which it is used.

The Annotation Generator has two pieces, the Annotations preference page and the Add Annotations dialog. The Annotations preference page enables you to create and store annotation templates. The Add Annotations dialog enables you to insert annotations into multiple files, using either stored annotation templates or custom written annotations.

Annotation Generation preference page

You can access the preference page by choosing Window > Preferences and then selecting the Progress OpenEdge > Editor > Annotations node.
<table>
<thead>
<tr>
<th>Annotation names</th>
<th>Lists all stored annotation templates. To browse through long lists, select a template and then drag the cursor up or down.</th>
</tr>
</thead>
</table>
| Annotation text  | Displays the text of the selected template. A template can include more than one annotation.  
The templates that ship with the Annotation Generator contain two specially coded preprocessors, %FILENAME% and %MODULE%. %FILENAME% expands to the filename including the extension. %MODULE% expands to the name of the directory where the file is stored. These are the only preprocessors the tool can expand. The tool does not support customer-created preprocessors. |
| Add              | Launches the New Annotation dialog.  
The New Annotation dialog does not check the syntax of the template you create. |
| Edit             | Launches the Edit Annotation dialog for the selected annotation.  
The Edit Annotation dialog does not check the syntax of the template you create. |
| Remove           | Removes the selected template from the internal template list. If you have exported your templates to a file, the template can be recovered by importing that file. |
| Import...        | Overwrites the current annotation templates with the contents of an external .dat file. |
| Export...        | Saves the current annotation templates to an external .dat file. |

Add Annotations dialog

You can access dialog by choosing **File > New > Other** and then selecting the **OpenEdge > Editor > Annotation Generation** node.

<p>| Select annotation template combo-box | Select an existing annotation template. The annotation text displays in the editor and can be customized. |</p>
<table>
<thead>
<tr>
<th><strong>Annotation text editor</strong></th>
<th>Displays the text of a selected annotation template. You can customize the annotation. You can also bypass the templates and type an annotation directly into the editor. You can type more than one annotation line into the editor. This enables you to write a set of standard annotations to multiple files in a single operation. NOTE: The editor does not check the syntax of the annotation.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Available Resource filter</strong></td>
<td>Enables you to filter the Available Resource treeview. Choose the Filter button to apply a filter.</td>
</tr>
<tr>
<td><strong>Available Resource treeview</strong></td>
<td>Displays all the resources in the current workspace, limited by any filter you applied. Check the toggle for all the files into which you want the annotation inserted. Checking a directory selects all its subordinate resources.</td>
</tr>
<tr>
<td><strong>Backup all selected resources toggle</strong></td>
<td>When selected causes files to backup before applying the annotation. By default, this option is turned on.</td>
</tr>
<tr>
<td><strong>Location</strong></td>
<td>The directory where the backup files are stored. NOTE: If you do not select a new backup location each time you run the tool, you might overwrite the original backup of a file.</td>
</tr>
<tr>
<td><strong>Finish</strong></td>
<td>Starts the Annotation Generator using the selected options.</td>
</tr>
</tbody>
</table>

**See also**
- ABL Annotations on page 822
- ABL Annotation content builder on page 831
- ABL annotation syntax on page 832

**Meta Catalog database**

Each catalog exists as a separate OpenEdge RDBMS database. The database is built and accessed through SQL commands.

When you have a large number of files in a catalog, you might want to consider the following options to increase performance:

- Start the catalog database with a higher value for the Blocks in Database Buffer (-B) parameter. The larger buffer allows more records to be in memory which reduces disk access.
• Start the catalog database with a higher value for the Minimum Clients per Server (-Mi) parameter. SQL servers can support multiple connections and reduce memory usage when this parameter is increased.

• Consider using Asynchronous Page Writers if there are a lot of updates to the catalog database.

A catalog database stores no data that cannot be extracted from the source code by the content builders. You can manually delete the database files and rebuild the same catalog from its definition. The content builders can extract two types of data:

• Meta data that you include in the source code as 4GL annotations.

• Data that can be read from the code without compiling it, information that you can parse from simple string comparison operations.

Because the content builders do not compile the source code, they cannot make substitutions. Content builders cannot expand preprocessors. Content builders cannot read and analyze the contents of included files. Content builders cannot expand a LIKE phrase to show the attributes a field inherits from another field. You might need to consider this behavior when creating search queries. For example, if you had to check all fields using the LONGCHAR data type, you might want the search query to also return all field records that have the LIKE attribute in the catalog.

A catalog database is specific to the workspace in which it is created. In order to add a source file into a catalog, you must first import that file into the workspace associated with the catalog.

Caution: The catalog database's schema is subject to change without notice in future releases. Any custom code created against the current schema might not function in future releases.

See also
Introducing the Meta Catalog on page 821
Local and master catalogs on page 823
Changing catalog definitions on page 826
Rebuilding a local catalog on page 828
Adding new files to a catalog on page 825

Meta Catalog Explorer view

The Meta Catalog Explorer view presents the data from all the catalogs defined in your workspace. You can access the view by choosing Window > Show View > Other and then selecting the Progress OpenEdge Meta Catalog > Meta Catalog Explorer node.

The left-hand pane contains a treeview to navigate through the data.

The right-hand pane contains a browse that displays details of the selected node, as follows:

<table>
<thead>
<tr>
<th>Property</th>
<th>Lists the database fields that describe the selected node.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>The value of the corresponding database field.</td>
</tr>
</tbody>
</table>

See also
Access to catalog data on page 822
Accessing the Meta Catalog Explorer on page 824
Using the Meta Catalog Explorer on page 830
ABL Annotation content builder on page 831
Meta Catalog preferences

You can access the Meta Catalog Preferences by choosing **Window > Preferences** and then selecting **Progress OpenEdge > Meta Catalog**. The Meta Catalog preferences consist of the following pages:

- **Catalog list** - A browse listing all the catalogs in your workspace.
- **Catalog Databases** - A tab folder for viewing and editing catalog definitions.
- **Content Builders** - A browse of all the content builders registered in your workspace.

**Catalog List page**

<table>
<thead>
<tr>
<th>Meta Catalog browse</th>
<th>Lists all catalogs currently defined in the current workspace. Shows each catalog's name, its update mode, whether it is a local or master catalog, and whether it is configured to start automatically.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong> A master catalog always shows &quot;false&quot; in the Auto start column, because it does not start when you open your workspace. If the administrator has not started a database server for it, you cannot access it.</td>
<td></td>
</tr>
<tr>
<td>New</td>
<td>Launches the Create Meta Catalog wizard that steps you through creating a new local catalog to your workspace.</td>
</tr>
<tr>
<td><strong>Note:</strong> Each catalog must have a unique name.</td>
<td></td>
</tr>
<tr>
<td>Add Master</td>
<td>Launches the Add Meta Catalog wizard that steps you through adding connection information for a master catalog to your workspace.</td>
</tr>
<tr>
<td><strong>Note:</strong> You cannot define a master catalog with the same name as a local catalog.</td>
<td></td>
</tr>
<tr>
<td>Remove</td>
<td>Removes a catalog definition from your workspace. If you choose a local catalog, you can choose to also remove the physical database files. If you choose a master catalog, you can only remove the definition.</td>
</tr>
<tr>
<td>Import</td>
<td>Imports a set of catalog and content builder definitions into your workspace. Any current definitions are discarded, even if there is no matching entry in the imported definitions. If a matching database is not found for an imported local catalog definition, you are given the option to rebuild the missing databases.</td>
</tr>
<tr>
<td><strong>Note:</strong> All the designated content builders run to recreate the content from the source code. This can take some time depending on the number of catalogs being built and the number of resources from which data is extracted.</td>
<td></td>
</tr>
</tbody>
</table>
### Export All

Exports the current set of catalog and content builder definitions to an XML file. By default, the definitions are stored in the catalog.xml file.

### Rebuild

Rebuilds the selected catalog. This option is disabled for master catalogs.

**Note:** This process deletes the physical database. All the designated content builders run to recreate the content from the source code. This can take some time if there are a lot of resources to extract data from.

---

#### Catalog Databases page

<table>
<thead>
<tr>
<th>Main page</th>
<th>Catalog Name combo-box</th>
<th>The catalog definition to view or edit. The label to the right of the combo box indicates whether it is a local or master catalog.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update for open projects only toggle box</td>
<td>If selected, when a rebuild is launched for this catalog, only the resources in open projects are run through the content builders.</td>
<td></td>
</tr>
<tr>
<td>Update catalog radio set</td>
<td>Method for updating a local catalog's content, as follows:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• On save of element - Appropriate content builders run each time a changed resource is saved. The data is written to all local catalogs whose definitions include the changed resource. This is the default setting.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• On rebuild - The catalog updates its contents only when it is rebuilt. You can do this with the Rebuild button on the Catalog List page or by using the command-line utility.</td>
<td></td>
</tr>
<tr>
<td>Restore Defaults</td>
<td>Loads the last catalog definition that was saved.</td>
<td></td>
</tr>
<tr>
<td>Apply</td>
<td>Saves any edits for the selected catalog to the internal version of the catalog.xml file in Progress Developer Studio for OpenEdge.</td>
<td></td>
</tr>
<tr>
<td>Database information</td>
<td>Local database toggle box</td>
<td>When selected, this indicates that the physical catalog database is on the local machine, rather than a remote machine. By default, local catalogs are assumed to be on the local machine.</td>
</tr>
<tr>
<td>-----------------------------------------------------------</td>
<td>---------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Meta Catalog database path</td>
<td></td>
<td>The full path of the physical catalog database, including the filename.</td>
</tr>
<tr>
<td>Service/Port for server (-S)</td>
<td></td>
<td>The connection port or service name for the catalog’s database server.</td>
</tr>
<tr>
<td>Additional server parameters</td>
<td></td>
<td>Any necessary startup parameters for the database.</td>
</tr>
<tr>
<td>Automatically start/stop server toggle box</td>
<td></td>
<td>Whether the catalog’s server should automatically start or stop when you enter or leave this workspace.</td>
</tr>
<tr>
<td>Connection information</td>
<td>Host</td>
<td>Name of the host where the catalog is.</td>
</tr>
<tr>
<td></td>
<td>Service/Port</td>
<td>The connection port or service for the catalog’s database server.</td>
</tr>
<tr>
<td></td>
<td>Database</td>
<td>The logical name of the database.</td>
</tr>
<tr>
<td></td>
<td>User</td>
<td>The user name to use connecting to the database.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> Because the catalog database is created with SQL, for local catalogs, you need to connect as a user with proper access to update the database.</td>
</tr>
<tr>
<td></td>
<td>Password</td>
<td>The password to use connecting to the database.</td>
</tr>
<tr>
<td>Mapping of root path</td>
<td>Root path</td>
<td>The root path for a resource as it was originally read into the catalog. If you have resources on different paths loaded into the same catalog, each root path displays on a separate line of the browse.</td>
</tr>
<tr>
<td></td>
<td>Local path</td>
<td>The path to your local copies of the resources in a master catalog.</td>
</tr>
<tr>
<td></td>
<td>Edit</td>
<td>Launches a dialog box where you can map a root path to a local path. When there is a mismatch between your local path and a master catalog’s root path for the same resource, setting this mapping between the paths produces the appropriate search results.</td>
</tr>
</tbody>
</table>
Lists the codes for content builders that are defined in this workspace. Checking the toggle box beside a code indicates that the content builder supplies data to this catalog.

<table>
<thead>
<tr>
<th>Content Builders</th>
<th>Code</th>
<th>Lists the codes for content builders that are defined in this workspace. Checking the toggle box beside a code indicates that the content builder supplies data to this catalog.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The description of the content builder.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Projects</th>
<th>Project name</th>
<th>Lists the names of all projects in this workspace. Checking the toggle box beside a project indicates that data from that project's resources should be added to this catalog whenever an appropriate content builder runs.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Description</td>
<td>The description of the content builder.</td>
</tr>
<tr>
<td></td>
<td>Class</td>
<td>The Java class for the content builder.</td>
</tr>
<tr>
<td></td>
<td>Extensions</td>
<td>The file extensions against which the content builder should run.</td>
</tr>
<tr>
<td></td>
<td>New...</td>
<td>Launches a dialog for entering a new content builder definition.</td>
</tr>
<tr>
<td></td>
<td>Edit...</td>
<td>Launches a dialog for editing a content builder definition. For example, you might add a custom file extension to the existing extensions.</td>
</tr>
<tr>
<td></td>
<td>Remove</td>
<td>Removes a content definition from this workspace. This does not delete the content builder class file.</td>
</tr>
<tr>
<td></td>
<td>Restore Defaults</td>
<td>Deletes the current definitions and replaces them with the default definitions of the predefined content builders that ship with Progress Developer Studio for OpenEdge.</td>
</tr>
<tr>
<td></td>
<td>Apply</td>
<td>Saves any edits for the selected content builder to internal version of the catalog.xml file in Progress Developer Studio for OpenEdge.</td>
</tr>
</tbody>
</table>

**Content Builders page**

<table>
<thead>
<tr>
<th>Code</th>
<th>The content builder's code.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The description of the content builder.</td>
</tr>
<tr>
<td>Class</td>
<td>The Java class for the content builder.</td>
</tr>
<tr>
<td>Extensions</td>
<td>The file extensions against which the content builder should run.</td>
</tr>
<tr>
<td>New...</td>
<td>Launches a dialog for entering a new content builder definition.</td>
</tr>
<tr>
<td>Edit...</td>
<td>Launches a dialog for editing a content builder definition. For example, you might add a custom file extension to the existing extensions.</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes a content definition from this workspace. This does not delete the content builder class file.</td>
</tr>
<tr>
<td>Restore Defaults</td>
<td>Deletes the current definitions and replaces them with the default definitions of the predefined content builders that ship with Progress Developer Studio for OpenEdge.</td>
</tr>
<tr>
<td>Apply</td>
<td>Saves any edits for the selected content builder to internal version of the catalog.xml file in Progress Developer Studio for OpenEdge.</td>
</tr>
</tbody>
</table>

**Note:** Changes to a content builder definition affect all catalogs in the workspace that use that content builder. You cannot change how a content builder behaves for a single catalog.

**See also**
- Introducing the Meta Catalog on page 821
- Local and master catalogs on page 823
- Changing catalog definitions on page 826
- Creating a local catalog on page 827
- Adding custom file extensions to content builders on page 824
- Setting Meta Catalog preferences on page 829
**Meta Catalog Search view**

You can access the Meta Catalog Search by choosing **Search > Search** and selecting the **Meta Catalog Search** tab.

<table>
<thead>
<tr>
<th>Projects to search</th>
<th>Lists all projects in your workspace. Select the projects whose contents you want to search.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catalogs to search</td>
<td>Lists all catalogs defined for your workspace. Select the catalogs whose contents you want to search.</td>
</tr>
<tr>
<td>Search criteria</td>
<td>Use this browse to construct the search query. All of the fields are combo-boxes that populate based on the projects and catalogs against which you are searching.</td>
</tr>
<tr>
<td>(</td>
<td>Enables you to insert up to 3 levels of parenthesis in your search criteria.</td>
</tr>
<tr>
<td>Content Type</td>
<td>Select an available content type. Content types match up with specific content builders. A content type is only available if one of the selected catalogs uses the corresponding content builder.</td>
</tr>
<tr>
<td>Entry Type</td>
<td>Select an available entry type. Entry types are classes of data extracted by a content builder. The available choices depend on what the content builders extracted from the source code and stored in the selected catalogs.</td>
</tr>
<tr>
<td>Entry</td>
<td>Select an available entry. An entry is a specific instance of an entry type. The available choices depend on what the content builders extracted from the source code and stored in the selected catalogs.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Select an available attribute. Attributes are classes of data that are children of certain entry types. The available choices depend on what the content builders extracted from the source code and stored in the selected catalogs.</td>
</tr>
<tr>
<td>Attribute Value</td>
<td>Select an available attribute value. An attribute value is a specific instance of an attribute. The available choices depend on what the content builders extracted from the source code and stored in the selected catalogs.</td>
</tr>
<tr>
<td>AND/OR</td>
<td>Enables you to insert either the AND or the OR logical operators in your search criteria.</td>
</tr>
<tr>
<td>)</td>
<td>Enables you to insert up to 3 levels of parenthesis in your search criteria.</td>
</tr>
<tr>
<td>Add</td>
<td>Adds a line to the bottom of the criteria in the browse.</td>
</tr>
<tr>
<td>Insert</td>
<td>Inserts a line above the currently selected line in the browse.</td>
</tr>
</tbody>
</table>
Removes the selected line in the browse.

Saved Criteria
Select a previously saved set of search criteria. The Search criteria browse will populate with the saved criteria. You can also type in a new name to save the current search criteria.

Saves the current search criteria under the name in the Saved Criteria combo-box.

Removes a saved criterion from the file.

Launches the search.
Note: The search first prompts you to ask if you want to save the search criteria. If you chose a named set of saved criteria as your starting point, you are given the choice to overwrite the saved criteria.

See also
Access to catalog data on page 822
Searching the Meta Catalog on page 829
Meta Catalog Explorer view on page 842

OECATALOG utility

The OECATALOG utility is used to administer a master catalog. You might use this command-line utility as part of the build process on the server hosting a source code management system. You can use the utility in a batch script or shell script on Windows, UNIX, and Linux platforms.

Full syntax

```
oecatalog -set |
  -build
  -catalog <configuration-file> [<catalog-name> ...]
  -root <root-path>[=<new-path>] [<root-path-2>[=<new-path-2>] ...]
  [-resource <resource-path> [<resource-path-2> ...]]
  [-project <project> [<project-2> ...]]
  [-type <content-type> [<content-type-2> ...]]
```

Where:

<configuration-file>

The catalog definitions file, usually called catalog.xml.

<catalog-name>

The name of a specific definition in the definitions file.

<root-path>

The root path of a project in the catalog.
<new-path>

The new root path where the project has been moved.

<resource-path>

The path, either absolute or relative to the -root, where the resource that you want to scan exists.

<project>

The project that you want to scan.

<content-type>

is the content type (ANNOTATION, APPSCHEMA, or ABLREFERENCE) for a content builder that you want to use on the resource.

---

**Note:**

- Use spaces, not commas, to delimit multiple arguments for an option.
- Use double quotes (" ") to escape spaces in paths.
- You must specify the -catalog and the -root option, and specify either the -set or the -build option.
- The -set option is used to change root paths in the catalog database.
- The -build option is used to update content in the catalog database.
- The -resource, -project, and -type options are optional.
- You can specify the -set, -build, -catalog, -root, -resource, -project in any order.
- You can specify multiple root paths. The projects are assumed to be the directories at the root path location.
- The -resource and -project options are equivalent if a relative path is used. The -project option can validate that the parameters are valid project names.
- If -resource or -project are not specified, then all the projects associated with the catalog in the definition file are processed.
- If -type is not specified, all the content builders associated with a catalog is used.
- When the -type option is specified, it overrides the association of content builders with the catalog. Since the -project and -resource option are equivalent, and it is similar to doing **Meta Catalog > Add** from within Progress Developer Studio for OpenEdge, a project or resource is added to a catalog only if the project is associated with the catalog.
- If you do not supply the full path in the -catalog option, the utility looks for the file in the local directory.

---

**Examples**

The following command would add content using all the content builders from all resources in C:\wrk and its subdirectories to all catalogs defined in the catalog definition file:

```
oecatalog -build -catalog "c:\eclipse\my workspace\catalog.xml" -root C:\wrk
```
The following command would only add content from the ABL Annotation content builder to the AnnoteCat catalog defined in the file:

```
oecatalog -build -catalog "c:\eclipse\my workspace\catalog.xml"
  AnnoteCat -root c:\wrk -type ANNOTATION
```

The following command would add content from a single project to all catalogs defined in the file:

```
oecatalog -build -catalog "c:\eclipse\my workspace\catalog.xml"
  -root c:\wrk -project myProject
```

The following command would change the root path to `C:\openedge\wrk` for all resources that were originally on `C:\wrk`:

```
oecatalog -set -catalog "c:\eclipse\my workspace\catalog.xml"
  -root "c:\wrk=c:\openedge\wrk"
```

**Note:** Because of the way Windows batch files handle equal signs (=), you should enclose any argument that contains an equal sign in double quotes (" ").

**See also**

Local and master catalogs on page 823
Building a master catalog on page 825
Introducing Tools for Business Logic

Tools for Business Logic in Progress Developer Studio for OpenEdge let you build models that represent ABL application modules as logical components. Using a graphical editor called the Component Designer, you diagram components, adding elements such as tables, columns, indexes, and relationships. As you build your components, OpenEdge stores them in a component model file or in a database.

An OpenEdge view called the Application Schema Explorer simplifies the process of finding and reusing existing code and model elements. This view lets you browse your code base and your existing model database for available elements.

After defining the logical structure of a component, you can generate the corresponding ABL source code. You can include this in an existing application, using the OpenEdge editor to modify it as necessary. You can also reverse this process, creating model components by extracting them from existing ABL source code.

For details, see the following topics:

- Concepts
- Tasks
- Reference
Concepts

Application modeling

In the context of Progress Developer Studio for OpenEdge, modeling is the process of creating visual representations of application components and their associated relationships, and storing these components in a format that makes it easy to share and reuse them. The tool used for this process is the Component Designer, a graphical editor.

With Tools for Business Logic, you can use two basic approaches:

- Model-to-code (forward engineering) - You begin by using the Component Designer to create a visual diagram of a component. Tools for Business Logic can generate the ABL code corresponding to the diagram on demand.

- Code-to-model (reverse engineering) - You use Tools for Business Logic to derive component diagrams from existing ABL code. You can then edit the diagrams with the Component Designer, and store and reuse the components.

The results of the design process are captured in two forms:

- A diagram, which is the visual representation of a single component and its subcomponents. This is the resource that you directly edit. The diagram file simply stores the information needed to render the picture that appears in the Component Designer.

- A component model, which stores detailed information about the structure and relationships of all components in a project. You assign each diagram to a specific component model, and although you may have many models in your workspace, each component belongs to only one model. When you save a diagram, or when you derive a diagram from existing code, its associated model is automatically updated. The component model is stored either in a OpenEdge database or as an XMI (EMF Ecore) file.

Two entities govern the creation of Tools for Business Logic components and the generation of the corresponding ABL code:

- Component definition file - An Ecore file that contains the rules of component structure. The component definition file determines what types of components you can create and what options are available in the Component Designer for constructing each type.

- Template - A JET (Java Emitter Template) file that specifies how the structure of a valid component is defined in ABL code. There is one template for each component type.

Progress Developer Studio for OpenEdge provides a standard model definition file and templates for the supported component types. Customization of these files requires advanced knowledge of Java and JET concepts and syntax, and working knowledge of EMF. If you choose to customize any of these files, be sure to keep backup copies of the original versions.

See also

Introducing Tools for Business Logic on page 851
Design output on page 855
Model storage options on page 856
Modeling new components on page 861
Generating code from a model on page 868
Creating diagrams from code on page 869
Design principles and the OpenEdge Reference Architecture

Tools for Business Logic offer the greatest benefit to those developers who adhere to the principles of the OpenEdge Reference Architecture (OERA), an application-design methodology based on maximizing re-usability and ease of maintenance by keeping data access, business logic, and user interface code in separate components. Progress Software Corporation encourages developers to become familiar with OERA concepts and to apply them to the extent possible in their own applications. However, you need not follow OERA guidelines to use the tools.

For links to more information about OERA go to the Architecture, SaaS, & Cloud Computing page on the Progress Communities Web site.

See also
Application modeling on page 852
Component types on page 853

The Business Logic tool set

Two Progress Developer Studio for OpenEdge tools are associated with Tools for Business Logic:

• Component Designer - The graphical editor with which you create and edit components.

• Application Schema Explorer - A specialized browsing view that helps you locate available building blocks for designing components.

In addition, the Tools for Business Logic perspective in its default state includes several views associated with other perspectives:

• DB Structure view - Lets you browse a database schema. You can drag and drop tables and fields into the Component Designer. (Associated with the DB Navigator perspective).

• Connections view - Lets you start and stop database sessions. In this view, you select the database to appear in the DB Structure view. The Connection Info view, showing detailed information about connected databases, is also included. (Associated with the DB Navigator perspective).

• Outline view - Lets you examine the structure of a model. You can drag existing elements from the model and drop them into the Component Designer for reuse. (Standard Eclipse view adapted for component models.)

• Problems view - Shows errors detected by the model validation process. (Standard Eclipse view.)

See also
Component Designer on page 870
Application Schema Explorer on page 871
Modeling components on page 861
Component Designer on page 870
Application Schema Explorer on page 871
Outline view for diagrams on page 872

Component types

You can use Tools for Business Logic to create and edit Temp tables and ProDataSets.
Temp tables
A temp table is a set of fields (columns) that temporarily stores multiple rows of data during a session. It generally exists only for the duration of the procedure that defines it, or at most for the duration of a session. A field in a temp table can be a direct copy of a field in a persistent database table, or it can be derived from other fields or data sources.

For information on defining and using temp tables, see OpenEdge Getting Started: ABL Essentials. You can find links to OpenEdge manuals on the OpenEdge Product Documentation Web site.

With the Component Designer, you can add the following elements to a temp table diagram:

- Columns
- Indexes
- Annotations
- Note
- Note links

See also
Application modeling on page 852
The Business Logic tool set on page 853
Modeling components on page 861
Component Designer on page 870

ProDataSets
A ProDataSet is a pre-defined view of data usually from multiple data sources (such as temp tables and database buffers) that are related to each other via data relationships. It is a potentially complex in-memory data structure that can be passed as a single parameter with a single handle from one procedure to another, within a single OpenEdge session or between sessions.

For information on designing, coding, and using temp tables, see OpenEdge Development: ProDataSets. You can find links to OpenEdge manuals on the OpenEdge Product Documentation Web site.

With the Component Designer, you can add the following elements to a ProDataSet diagram:

- Temp tables
- Buffers
- Columns
- Indexes
- Relationships
- Annotations
- Notes

See also
Temp tables on page 854
Application modeling on page 852
Design output

Creating a component in the Component Designer yields two forms of stored output:

- A component diagram.
- An application model.

**Note:** You must explicitly generate the ABL source code for your application. Code is produced on demand, not automatically.

Component diagrams

The diagram is the resource that you directly edit with the Component Designer. It is stored as a file with an extension of .dgm.

Each diagram represents a single component and its subcomponents. For example, a ProDataSet diagram can include any number of temp tables and buffers, but only one ProDataSet. The diagram may also include hierarchical relationships between a component and its subcomponents, and two-way (peer-to-peer) relationships between elements.

The diagram file stores only information about the picture displayed on the Component Designer canvas, such as size, position, and color of the elements. It does not contain any schema definitions or application logic; that information is stored in the component model.

**See also**
- Component models on page 855
- Modeling components on page 861

Component models

The component model stores the logic and schema definitions for all components in a project. When you create a component, you assign it to a model. Each time you save a component diagram, that component's model is automatically updated.

While you are editing a diagram, the Outline view displays the structure of the associated models, allowing you to browse the contents of all components. You can reuse components, either within one model or in separate models, by dragging them from the Outline view to the canvas.

You have a choice between two model storage formats:

- XMI file (EMF Ecore file format)
- OpenEdge database

Be sure to read the next topic, "Model storage options," to understand the advantages and limitations of each of these formats.

**See also**
- Model storage options on page 856
Model storage options

You must choose one of two formats for storage of your component models:

- XMI (EMF Ecore file)
- OpenEdge database

You set your own preference for your local workspace, a global setting that applies to all projects. In a typical collaborative environment, all users should select the same format. This topic explains the implications of both choices.

XMI storage

XMI is an XML-based file format that is gaining acceptance as a standard application modeling persistence and transport mechanism (see http://www.eclipse.org/emf/ for more information). Advantages of this format include:

- You can maintain multiple model files and assign each individual components to the model of your choice.
- The file format is readily usable with UML modeling tools.

Limitations of the XMI format include:

- Because the model is a flat file, only one user can have write access to it at any given time.
- Complex models result in very large files that can be relatively difficult to share.
- The Application Schema Explorer does not support the display of XMI model content. If using XMI storage, you must generate source code, and update a Meta Catalog with that code, before you can see the content in the Application Schema Explorer. (See the Meta Catalog volume in Progress Developer Studio for OpenEdge help.)

Database storage

If you use database storage, you are limited to a single model. All components that you create belong to that model. However, database storage offers the following advantages:

- Multiple users can access the model concurrently.
- The Application Schema Explorer displays the model content with no requirement to generate code.

See also

- Application modeling on page 852
- Setting Tools for Business Logic preferences on page 857
Tasks

Setting Tools for Business Logic preferences

Tools for Business Logic preferences determine your model storage format and certain characteristics of the Component Designer. In addition, if you select database storage, you can create a new model database or specify an existing one. Preference settings apply to all projects in your workspace.

To view or edit preference settings, select Window > Preferences. In the left pane of the Preferences window, expand Progress OpenEdge and then Tools for Business Logic.

Choosing a background color

You can control the background color of the Component Designer canvas:

1. Select Window > Preferences.
2. In the left pane of the Preferences window, expand Progress OpenEdge and select Tools for Business Logic.
3. Click the Diagram background color button to open a color palette, select the desired color, and click OK.

See also
The Business Logic tool set on page 853
Setting Tools for Business Logic preferences on page 857
Tools for Business Logic options on page 873

Controlling automatic linking

You have an option to have the Component Designer automatically create default relationships between certain components. For example, when you add a temp table to a ProDataSet diagram, a "contains" relationship, linking the ProDataSet to the table, can be automatically created.

This option is enabled by default. To change the setting:

1. Select Window > Preferences.
2. In the left pane of the Preferences window, expand Progress OpenEdge and select Tools for Business Logic.
3. Click the Auto link components check box to toggle the option off or on.

See also
The Business Logic tool set on page 853
Adding relationships on page 866
Setting Tools for Business Logic preferences on page 857
Tools for Business Logic options on page 873
Determining how models are stored

You can store your component model in either of two formats:

- XMI (Ecore EMF) file
- DB (OpenEdge database)

The default setting is XMI.

If you select XMI model storage, you can choose whether to use advanced features:

- Advanced features disabled (default) - When you create components, you are not prompted to specify a model. All components are stored in a single model file named appModel.t4bl, located in the project root folder.
- Advanced features enabled - When you create components, you specify one or more model files. You control the name and location of your models.

To set model storage options:

1. Select Window > Preferences.
2. In the left pane of the Preferences window, expand Progress OpenEdge and select Tools for Business Logic.
3. Select either XMI or DB in the Persistence area.
4. Click the Disable advanced wizard features check box to toggle advanced features off or on.

See also

Model storage options on page 856
Creating a model database on page 858
Editing database settings on page 860
Tools for Business Logic options on page 873

Model database management

This section explains how to create a model database, edit its settings, and share connection information.

Creating a model database

If you choose the database storage option for your models, you must create a model database on your local machine, or specify a remote database:

1. Start the Create Component Model Database wizard by either of two methods:
   - Select File > New > Other > OpenEdge > Tools for Business Logic > Create Component Model Database.
   - Select Window > Preferences. In the left pane of the Preferences window, expand Progress OpenEdge and then Tools for Business Logic, and select Component Model Database. Then click New.

2. In the Component Model Database dialog, click New.
3. Enter a logical name for the database and click Next.
4. If creating a new database on your local machine, click Create new database to make the fields writable.
Note: If specifying a remote database, leave the Create new database field unchecked, and skip to step 10.

5. In the Component model database path field, enter the complete file path to the .db file that you want to create. You can either type the path and file name; or, click Browse, navigate to the directory where you want to create the database, enter a file name with a .db extension, and click Save to insert the path.

6. Enter a service name or port number for the database server.

7. Enter any additional parameters for the database server.

8. If you want the database server to start automatically when you start Progress Developer Studio for OpenEdge and to stop when you quit, click the check box.

9. If you want to create the database with default settings, click Finish. OpenEdge creates the database, assigning administrator privileges to your login name, with no password. No projects are associated with the database. You can associate projects and, if necessary, change the login values in the Database Settings dialog.

10. Click Next if any of the following apply:

   • You are specifying a remote database.
   • You want to add a password for the administrator's login to a local database.
   • You want to specify projects for which component models will be stored in the database.

Note: If you clicked Next in the preceding step, you see fields for database connection settings. Any values that you entered on the preceding page appear. If you are creating a new local database, you need not make any changes on this dialog. Your login name is proposed by default as the DBA user; you can optionally change this User value or add a DBA password. If you specified no values on the preceding page, enter a remote host name, service name or port number, a valid database user name, and the user's password if necessary.

11. Click Finish if you want to create the database without associating projects, or click Next to specify projects for which component models will be stored in the database.

12. If you clicked Next in the preceding step, you see the Tools for Business Logic Project Association dialog. To associate all projects with this database, click Add All. To associate projects selectively, click those projects for which you plan to store component models, and click Add. Then click Finish to complete the database creation process.

Note: Progress Developer Studio for OpenEdge does not support the use of multiple component model databases. You can create only one per workspace.

See also
Model storage options on page 856
Editing database settings on page 860
Setting Tools for Business Logic preferences on page 857
Create Component Model Database wizard on page 876
Sharing remote database settings

If two or more people need to use the same model database that is located on a remote host, exporting and importing the settings simplifies the process of setting up connection parameters and project associations on multiple machines.

1. To export database information:
   a) Define the remote connection parameters, and optionally the project associations, as explained in Creating a model database.
   b) Select Window > Preferences > Progress OpenEdge > Tools for Business Logic > Component Model Database.
   c) Click Export.
   d) At the Export Component Database file browser, navigate to the location where you want to save the exported setting, enter a file name with a .xml extension, and click Save.

   Note: Other developers can then import the exported settings.

2. To import exported database settings:
   a) Select Window > Preferences > Progress OpenEdge > Tools for Business Logic > Component Model Database.
   b) Click Import.
   c) At the Import Component Database file browser, navigate to the location of the exported .xml file. Select the file and click Open.

See also
Model storage options on page 856
Creating a model database on page 858

Editing database settings

After creating a component model database, you may need to review or modify some of the settings that you used when you created it. To do so:

1. Select Window > Preferences > Progress OpenEdge > Tools for Business Logic > Component Model Database > Database Settings.
2. Click the Database information, Connection information, and Projects tabs to view the settings and make any desired changes.
3. Click Apply or OK to save any changes.

See also
Model storage options on page 856
Creating a model database on page 858
Database settings on page 875
Modeling components

This section explains the process of creating Business Logic components.

Creating a new component

1. Complete the following steps to create a new Business Logic component:
   a) Select **File > New > Other > OpenEdge > Tools for Business Logic > Business Logic Component** and click **Next**.
   b) Enter a folder in your workspace as the location for the component diagram.
   c) In the Component name field, enter a file name with a .dgm extension.
   d) Select one of the valid component types.
   e) Click **Finish** or **Next**, whichever button is enabled.

   **Note:** If you click **Finish**, the diagram file opens in the Component Designer and is associated with your model database or with the default appModel.t4bl file, as applicable. You can proceed to edit the diagram. If you click **Next**, a dialog opens where you must specify a component model file as explained in the next step.

2. You must populate the list of T4BL model files with one or more file names. Your new component will be added to the model at the top of the list. If more models are listed, they will be included in the Outline view, from which you can drag components to the diagram of the component that you are creating. Use this feature to associate the component with related models to which you want to have access while building the component.

3. Use one or more of the following methods to populate the model list:
   a) Click **Create**. Then specify the name and location of a new .t4bl file, and click **OK**.
   b) Click **Browse Workspace** to open a browser for your current workspace. Navigate to and check one or more folders or .t4bl files (check a folder to select all .t4bl files in that folder and its subfolders; click the folder name to view and select individual .t4bl files contained in that folder). Click **OK** to add all selected .t4bl files to the list.
   c) Click **Browse FileSystem** to open a standard file browser with a *.t4bl filter. Navigate to and select a .t4bl file anywhere on your file system, and click **Open** to add it to the list.

4. To remove a model from the list, select it and click **Remove**.

5. When the list shows the model files in the desired order, click **Finish** to create the diagram file that you specified in step 2. The file opens in the Component Designer, ready to have subcomponents or other elements added.

See also

- Component types on page 853
- Application modeling on page 852
- The Business Logic tool set on page 853
- Design output on page 855
- Editing a component on page 862
- Creating diagrams from code on page 869
Editing a component

The topics in this section explain how to define the content of a component.

Adding new data structures

To add a new data structure to the component you are editing:

1. Click an item in the New Components drawer of the palette.
2. Move the cursor to the area of the canvas where you want to place the item, and click again.
   Depending on the component type, you must place it either inside a component rectangle (for example, when adding a column to a table) or on a blank area of the canvas (for example, when adding a temp table to a ProDataSet.) The cursor appears as an arrow when it is over a valid area to place the component.
3. New components appear in the diagram with default names. To change the name of a component, click the name once to select it, click it again to make it editable, and type the new name. Alternatively, select the component whose name you want to change, and edit its Name value in the Properties view.

Editing indexes

In the Properties view for an index component, you can specify the columns to be indexed and other attributes. To edit the values:

1. Select the index in the diagram.
2. Open the Properties view and click in the Value column to the right of IndexProperties (under Attributes). An ellipsis button (⋯) appears at the right side of the column. Click this button to open the Index Properties dialog.
3. Enter the index name in the field at the top.
4. Either click Word, or click one or both of Primary and Unique, to specify the type of index.
5. Select one or more columns in the Available Columns list on the left and click Add to add them to the index and move them to the Selected Columns list on the right.
6. Select columns on the right and use the Ascending and Descending radio buttons to specify sorting order; use the Move Up and Move Down buttons to change the sequence.

See also

The Business Logic tool set on page 853
Using existing data structures on page 862
Adding relationships on page 866
Adding diagram notes on page 866
Adding annotations on page 867
Deleting components on page 867
Component Designer on page 870

Using existing data structures

As you build new components, Progress Developer Studio for OpenEdge tools let you easily copy or reuse existing data structures from several sources. You can copy and reuse data sets, tables, and columns from a component model, a database, or source code.
See also
- The Business Logic tool set on page 853
- Identifying application schema elements on page 865
- Component Designer on page 870
- Outline view for diagrams on page 872

Copying from a component model

You can use the Outline view or the Component Designer palette to reuse elements found in a component model in your workspace.

**Note:** In both of these cases, reusing a table results in a proxy object appearing in your diagram. You cannot modify a proxy object; any changes must be made to the source object. A proxy object appears in the diagram with a broken-line border rather than the solid border that signifies an editable object. If you want to edit the source object, right-click the proxy and select Open Diagram from the menu. The source object diagram opens in the Component Designer, and you can edit it. After you save your changes, they are reflected in the proxy object.

1. To use the Outline view:
   a) In the Component Designer, open the diagram to which you want to add existing elements.
   b) In the Outline view, browse to locate the desired element.
   c) Select the element, hold down the left mouse button, and drag to the area of the canvas where you want to place the item. Then release the mouse button.

   **Note:** Depending on the component type, you must place it either inside a component rectangle (for example, when adding a column to a table) or on a blank area of the canvas (for example, when adding a temp table to a ProDataSet.) The cursor appears as an arrow when it is over a valid area to place the component.

2. To use the palette:
   a) Click an item in the Existing Components drawer of the palette.
   b) Move the cursor over the canvas. A filtered instance of the Outline view appears, showing all available elements of the selected type. To find a specific element, type the first one or more characters of its name in the Name field at the top of the window; the view shows only those elements that begin with that string.
   c) Select the element that you want to use, and click OK. The view window closes.
   d) Move the cursor to the area of the canvas where you want to place the item, and click again.

   **Note:** Depending on the component type, you must place it either inside a component rectangle (for example, when adding a column to a table) or on a blank area of the canvas (for example, when adding a temp table to a ProDataSet.) The cursor appears as an arrow when it is over a valid area to place the component.

See also
- The Business Logic tool set on page 853
Copying from a database

To copy from a database, you can add components to a diagram from the DB Structure view:

1. In the Connections view, connect to the database from which you want to copy data structures.
2. In the DB Structure view, browse to the desired table; expand the table node and the Columns node below it to display fields.
3. Select the element, hold down the left mouse button, and drag to the area of the canvas where you want to place the item. Then release the mouse button.

Note: For more information about the DB Structure view, see the Database Navigator volume in Progress Developer Studio for OpenEdge help.

Copying from source code

To copy from source code, add components to a diagram from the source code that defines them, by dragging elements from the code outline to the Component Designer.

Note: If you want to copy a component defined in code to a diagram of the same component type (for example, copy a temp table to a new temp table or a ProDataSet to a new ProDataSet), you must complete the procedure below before adding any other elements to the new component.

1. To use this technique, you need to split the editing area so that it shows both the source file and the diagram. You do not need to view the source code in the Editor, but you need to put editing focus on the source file without blocking the diagram from view.
   a) Open both files.
   b) Point to the tab of the currently visible file, hold down the left mouse button, and drag toward the bottom of the editing area until a horizontal line appears bisecting the editing area. Then release the mouse button. You now see both editor instances.
   c) You can click and drag the border between the two editing panes to increase the size of the Component Designer pane.
   d) Put focus on the source file by clicking its tab. The structure of the code now appears in the Outline view.
2. In the Outline view, browse to locate the desired element.
3. Select the element, hold down the left mouse button, and drag to the area of the canvas where you want to place the item. Then release the mouse button.
Note: Depending on the component type, you must place it either inside a component rectangle (for example, when adding a column to a table) or on a blank area of the canvas (for example, when adding a temp table to a ProDataSet.) The cursor appears as an arrow when it is over a valid area to place the component.

See also
The Business Logic tool set on page 853
Identifying application schema elements on page 865
Component Designer on page 870
Outline view for diagrams on page 872

Identifying application schema elements
You can use the Application Schema Explorer to locate schema elements that do not exist in the physical database but rather are created at runtime by program execution. These application schema elements include:

- ProDataSets
- Temp tables
- Fields defined in temp tables

Application schema elements appear in the Application Schema Explorer if they are defined in either or both of two places:

- code that is indexed in an OpenEdge Meta Catalog accessible to your workspace. The Application Schema Explorer includes available schema elements from all of your local catalogs and any master catalogs that you link to. (See the OpenEdge Meta Catalog help volume for more information.)
- Your application model database. (The Application Schema Explorer does not support the display of models stored in XMI format.)

Using the Application Schema Explorer
1. To open the Application Schema Explorer, select Window > Show View. In the Show View window, expand the OpenEdge Tools for Business Logic folder and select Application Schema Explorer.
2. The tree in the Application Schema Explorer has two expandable top-level nodes, Catalogs and Models. Expand these nodes to display nodes for your available catalogs and models.
3. Expand any catalog or model node and its sub-nodes to see schema elements that are available in each category.
4. To determine the location of an element, use the information in the node labels and in the property viewer in the right pane.
5. After locating an element in the Application Schema Explorer, you can reuse in a component diagram by copying it from:
   - The code Outline view for catalog-based elements.
   - The model Outline view for model-based elements.

See also
Using existing data structures on page 862
Application Schema Explorer on page 871
Outline view for diagrams on page 872
## Adding relationships

You can add links between components in a diagram to indicate relationships.

**Note:** By default, certain links are automatically created when you add components. For example, when you add a temp table to a ProDataSet diagram, a hierarchical "contains" link is created. You can disable automatic link creation in Tools for Business Logic Preferences.

1. Create a diagram or edit an existing one.

2. In the **Connections drawer of the Component Designer** palette, click **Two-way Reference**.

3. In the diagram, click one component, and then click the component to be linked. A broken line appears between the two components, with a text label identifying the relationship type.

   - **Relationship names**: Each relationship has a name, by default the same as the relationship type. If you want to change the default name, select the link on the diagram and then open the **Properties view**. Click twice on the **Name** value in the right column, making the field editable, and enter the name of your choice. This name now appears on the diagram, preceded by a plus sign (+).

   - **Adjusting link lines**: You can alter the shape and position of the line that depicts a relationship. Select the line and then position the cursor over the break node that appears at the midpoint. When the cursor appears as a cross, you can click and drag to reshape the line as desired. Each time you move a break node, a new node appears on either side of it, making it possible to shape the line in any way you wish.

**Note:** The component type definition determines the types of elements that can be linked, and you cannot place a link that is not allowed by the type definitions of both elements. In the case of a hierarchical relationship such as that between a data set and a table, you must click first on the parent element and then on the child. For example, you must click first on the ProDataSet and then on the temp table.

### See also

- [Component types](#) on page 853
- [Controlling automatic linking](#) on page 857
- [Component Designer](#) on page 870

## Adding diagram notes

You can add notes to a diagram for clarification or any other purpose. Notes appear only on the diagram; they are not stored in the model or in generated code.

1. Create a diagram or edit an existing one.

2. In the **Other drawer of the Component Designer** palette, click **Diagram Note**.

3. Click on the canvas where you want to place the note.

4. Click once in the note rectangle to make the text editable, and enter the text of the note. You can resize the rectangle by dragging its borders.

5. Optionally, you can add a link to associate the note visually with a component in the diagram. Click **Note Link** in the **Relationships drawer** of the **Component Designer** palette. Click the note first, and then click the component that you want to link the note to. A broken line connects the two entities.
Adding annotations

You can add annotations to a temp table or ProDataset component in a diagram so that the annotations will be included in the generated code.

Annotations are used to store metadata that allows the source code to be indexed in one or more catalogs. See the Meta Catalog volume in Progress Developer Studio for OpenEdge help for detailed information about catalogs and annotations.

- To add an annotation to a component:
  a) Create or edit a diagram.
  b) In the Other drawer of the Component Designer palette, click **New Annotation**.
  c) Click on the temp table or ProDataset to which you want to add the annotation.

The annotation is added to the diagram with placeholder values for the annotation name and the name-value pair for a single attribute. Annotation syntax is:

```const
@annotation_name [(attribute = "value", attribute = "value")...]
```

- To edit the values:
  a) Select the annotation in the diagram.
  b) Open the **Properties view** and click in the Value column to the right of Annotation Properties (under Attributes). An **ellipses** button (・・・) appears at the right side of the column. Click this button to open the **Annotation Properties** dialog.
  c) Enter the annotation name in the field at the top.
  d) Click the text in the **Key column** and enter the attribute name.
  e) Click the text in the **Value column** and enter the attribute value.
  f) Use the **New entry and Remove entry** buttons to add more attributes or to delete previously entered attributes.
  g) Click **OK**.

See also

**Generating code from a model** on page 868

Deleting components

When you delete diagram components, you can choose whether to delete them from both the model and the diagram, or from the diagram only.

To delete components:

1. Select one or more components.
2. To delete the selected components from the diagram only, press the Delete key.
3. To delete the components from both the model and the diagram, press Shift +Delete.
**Matching diagrams to ABL code**

You can generate the source code that defines the components in a diagram. You can also create diagrams from existing source code.

**Validating a component diagram**

You can validate a component to make sure no errors, such as missing required attributes, will prevent generated source code from being usable.

To validate a component:

1. Edit the component diagram in the Component Designer.
2. Right-click on the canvas and select **Validate Component** from the menu.
3. You see either a success message or a failure message. If validation fails, look at the Problems view to see what errors occurred.

**Generating code from a diagram**

You can produce source code that defines the model represented in a component diagram.

By default, Progress Developer Studio for OpenEdge generates separate include files for a ProDataSet and for each temp table contained in a ProDataSet component. Each include file has the same name as the corresponding component, and a .i extension. You can change this behavior by editing the component's property settings.

- To generate code from a diagram:
  a) Edit the component diagram in the **Component Designer**.
  b) Right-click on the canvas and select **Generate Code** from the menu.
  c) Progress Developer Studio for OpenEdge validates the component, and if no errors are found, creates the source files in the same folder as the diagram. These files open in the Editor.

If validation fails, you have the option of canceling or proceeding with the code generation. Look at the Problems view to see what errors occurred.
• To customize code generation, in the **Properties view**, you can edit the Code Generation settings for a component to control the following aspects of code generation:
  
  • Location of generated file - Edit the ContainerName property.
  • File name - You can specify a prefix to be added to the name of the diagram. Enter a value for the FileNamePrefix property.
  • Extension - Edit the FileExtension property.
  • Single file versus separate files - For a component contained in a parent component, such as a temp table in a ProDataSet, you can generate a separate file or include the code in the definition of the parent component. Edit the GenerateSeparateFile property.

**See also**

Validating a component diagram on page 868
Creating diagrams from code on page 869
Component Designer on page 870

**Creating diagrams from code**

Progress Developer Studio for OpenEdge can extract component definitions from existing source code and create the corresponding component diagrams. Use the following procedure:

1. In the **Resources view**, right-click the source code file.
2. From the menu, select **Tools for Business Logic** > **Code to Model**. The **Code to Model Component** wizard starts.
3. The wizard shows all eligible components found in the source code, and proposes the source file as the location for the new diagrams. For each component, you can:
   
   a) Check or clear the box in the **ObjectName column** to specify whether you want a diagram created for the component.
   
   b) Edit the name of the diagram file.

   **Note:** You can also control where the diagrams are created by changing the directory path in the Location field.

4. Click **Finish** to create the selected component diagrams. The diagrams open in the Component Designer

**See also**

Component diagrams on page 855
Generating code from a diagram on page 868
Code to Model Component wizard on page 872
Reference

Component Designer

The Component Designer is the primary tool for developing application models. It opens when you create or edit a diagram (.dgm file).

The Component Designer features:

- An editing canvas that displays a diagram of the component and subcomponents that you are editing. Rectangles represent the components and contain text labels indicating their content. Lines connecting components represent relationships.

- A palette from which you select elements to add to the diagram. The list of available elements depends on the model definition for the component type. You click on the desired element type and then on the canvas to place the element.

The visual representation that you create is stored as a diagram file with a .dgm extension. Each time you save a diagram, the logical elements that it depicts are automatically stored in the associated model.

Using the palette

By default, the palette is at the right edge of the Component Designer window and is closed. It opens when you position the cursor over the Palette bar, stays open as long as the cursor is inside the palette area, and closes when you move the cursor outside the area.

If you prefer to keep the palette open, click the left arrow (стрелка левая) at the top of the Palette bar. The symbol becomes a right arrow (стрелка правая), which you can click to close the palette again.

To move the palette to the opposite side of the Component Designer window, open it, move the cursor to the title bar at the top, left-click and drag across the window, and release the mouse button.

Elements that you can add from the palette are organized in labeled sections called drawers. You open a drawer by clicking its label (for example, New Components). By default, only one drawer is open at one time; if you open a new drawer, the other closes. To keep a drawer open, click the pin symbol (запертое окно) at the right edge of the label; click the pin again to let the drawer close.

Selection tools

At the top of the palette are two cursor options that offer different methods for selecting objects on the canvas:

- Select - Lets you select single objects by pointing and clicking, or multiple objects, one at a time, by shift-clicking. When this tool is active, the cursor appears as an arrow.

- Marquee - Lets you drag a temporary rectangle to enclose the objects that you want to select. When this tool is active, the cursor appears as a pair of crosshairs.

Click to choose one of these cursors. When you click an element in a palette drawer, the select cursor is automatically activated.

Placing elements on the canvas

To add an element to the diagram, click its label (for example, New Column) in the drawer, and then click the area of the canvas where you want to place it. Some elements must be placed inside existing elements on the canvas; for example, a column must be placed inside a table. Others must be placed outside any other elements; for example, a table must be placed directly on a blank area of the canvas.
The appearance of the cursor indicates whether an element can be placed at the current position. An arrow with a plus sign indicates that the element can be placed. A circle with a diagonal bar through it indicates an ineligible area of the canvas.

If you click an element in a drawer and then decide not to place it, press Esc or click another element.

You can reposition a rectangular object, such as a table, on the canvas. Select one or more objects by clicking or shift-clicking any non-text area of the rectangle with the arrow cursor, or by enclosing them with the marquee cursor. Then, using the select cursor, click and drag the selection to the desired position.

See also
- The Business Logic tool set on page 853
- Modeling new components on page 861

Application Schema Explorer

The Application Schema Explorer helps you locate existing logical elements that you can reuse in new models. For example, it can show you all available ProDataSets.

The Application Schema Explorer appears when you select Window > Show view > Other > OpenEdge Tools for Business Logic > Application Schema Explorer.

The Application Schema Explorer includes:

- A tree-style browser that groups valid element types under separate nodes that you expand to show available elements. You can expand and collapse nodes by clicking the + and - symbols.
- A property viewer that displays useful information about the selected tree node.

To find the location of a specific element, such as a temp table, select its node in the browser and note the path properties in the viewer.

Application Schema Explorer content

This view displays information about elements found in one or both of the following sources:

- Meta catalog - If you have one or more local or master catalogs configured for your workspace, all eligible elements defined in the indexed code base are available under the Catalogs node of the browser. See the Meta Catalog volume in Progress Developer Studio for OpenEdge help for more information.
- Component model database - If you use the database storage option for your component models, all eligible elements that you create in the Component Designer are available under the Models node of the browser as soon as you save the diagram. The Application Schema Explorer does not support the display of models stored in XMI format.

Note: If you use XMI model storage, only the Catalogs node shows content. The components that you create with Tools for Business Logic are added to catalogs only after you generate ABL code from your component diagrams.

See also
- Model storage options on page 856
- Application modeling on page 852
- Component types on page 853
- Identifying application schema elements on page 865
- Generating code from a model on page 868
Outline view for diagrams

When the Component Designer is active (that is, you are editing a .dgm file), the Outline view displays the contents of one or more component models, as follows:

- The model that actually contains the component you are currently editing. If you are using database storage, the outline contains only one model.

- If you are using XMI storage, any other models that you associated with the component when you created it.

The Tools for Business Logic perspective includes the Outline view by default. To display it if is not open, select Window > Show view > Basic > Outline.

The Outline shows a first-level node for each model. If multiple model files are shown, the one that contains the component you are editing appears first. To expand a node to display its content, click the dot to the left of the label at the right angle of the vertical and horizontal branch lines (i.e., Expand nodes to browse the model structure.

You can drag elements from the outline to the canvas to add them to the current component, provided the context allows it. For example, you can drag a table to a ProDataSet, or a column into a temp table.

See also

The Business Logic tool set on page 853
Using existing data structures on page 862
Modeling components on page 861
Component Designer on page 870

Code to Model Component wizard

The Code to Model Component wizard lets you select from a list of component definitions found in an ABL source file the components for which you want to create diagrams. The wizard opens when you right-click a source file in the Resources view and select Tools for Business Logic > Code to Model.

The following fields and commands are available:

<table>
<thead>
<tr>
<th>Fields</th>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource</td>
<td>The source code file selected in the Resources view.</td>
<td></td>
</tr>
<tr>
<td>Location</td>
<td>The folder in which diagram (.dgm) files are to be created.</td>
<td></td>
</tr>
<tr>
<td>Object Name</td>
<td>The component name as defined in the source file. Use the check box preceding the name to select or deselect the component for diagram creation.</td>
<td></td>
</tr>
<tr>
<td>Component Type (read-only field)</td>
<td>The type of Business Logic component.</td>
<td></td>
</tr>
<tr>
<td>Component Name</td>
<td>The name of the diagram file to be created (editable).</td>
<td></td>
</tr>
</tbody>
</table>
### Commands

<table>
<thead>
<tr>
<th>Commands</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browse</td>
<td>To open a standard file browser that you can use to specify a location for the diagram files.</td>
</tr>
<tr>
<td>Select All</td>
<td>To select all components.</td>
</tr>
<tr>
<td>Deselect All</td>
<td>To deselect all components.</td>
</tr>
<tr>
<td>Finish</td>
<td>To create diagrams for the selected components.</td>
</tr>
<tr>
<td>Cancel</td>
<td>To close the wizard without creating any new components.</td>
</tr>
</tbody>
</table>

### See also

- [Component diagrams on page 855](#)
- [Component types on page 853](#)
- [Configuring a JDBC driver on page 852](#)
- [Creating diagrams from code on page 869](#)
- [Creating a new component on page 861](#)

### Tools for Business Logic options

This section explains the pages of the **Preferences** dialog. To access these settings, select **Window > Preferences > OpenEdge > Tools for Business Logic.**

### Component Designer options

These options govern basic behavior of the Component Designer. You open this dialog by selecting **Window > Preferences > Progress OpenEdge > Tools for Business Logic.**

The following fields and commands are available:

<table>
<thead>
<tr>
<th>Fields</th>
<th>Component Directories</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Not currently used; do not modify.</strong></td>
</tr>
<tr>
<td>Persistence</td>
<td>Radio buttons for selecting the storage format for component models.</td>
</tr>
<tr>
<td>Auto-link components</td>
<td>An option to have the Component Designer automatically create default relationships between certain components.</td>
</tr>
<tr>
<td>Disable advanced wizard features</td>
<td>An option to skip selection of a model file when creating a new component with XMI storage in effect. If this option is selected, the component is stored in the default appModel.t4bl file in the project root folder.</td>
</tr>
<tr>
<td>Diagram background color</td>
<td>An option to select a color for the Component Designer canvas.</td>
</tr>
</tbody>
</table>
See also
Design output on page 855
Setting Tools for Business Logic preferences on page 857
Adding relationships on page 866
Component Designer on page 870

Database list

This dialog identifies your component model database, if any. You open the dialog by selecting Window > Preferences > Progress OpenEdge > Tools for Business Logic > Component Model Database.

The following fields and commands are available:

<table>
<thead>
<tr>
<th>Fields</th>
<th>Component database name</th>
<th>The logical name for the database.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Auto-start</td>
<td>The setting (true or false) of the option to start the database server automatically when you launch Progress Developer Studio for OpenEdge and stop it when you quit.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Commands</th>
<th>New</th>
<th>Starts the Create Component Model Database wizard (available only when no model database is currently defined).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Remove</td>
<td>To delete the model database from disk, if it is local to your machine, and disassociate it from your component models.</td>
</tr>
<tr>
<td></td>
<td>Import</td>
<td>To read settings for a remote database from an exported XML file and apply them for use with your projects, replacing any settings currently in use.</td>
</tr>
<tr>
<td></td>
<td>Export</td>
<td>To create an XML file containing your current database settings, allowing other users to import the file and apply the same settings for a shared database.</td>
</tr>
<tr>
<td></td>
<td>OK</td>
<td>To save all current values and close the Preferences dialog.</td>
</tr>
<tr>
<td></td>
<td>Cancel</td>
<td>To close the Preferences dialog without saving any changes.</td>
</tr>
</tbody>
</table>

See also
Model storage options on page 856
Model database management on page 858
Database settings

The **Database Settings** dialog shows the current information about the name, location, connection parameters, and project associations of the component model database, if defined, and lets you make changes to these settings. This dialog appears when you select **Window > Preferences > Progress OpenEdge > Tools for Business Logic > Component Model Database > Database Settings**.

Settings appear on three tabs: Database information, Connection information, and Projects.

**Database information tab**

The following fields are available:

<table>
<thead>
<tr>
<th>Fields</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component database name (read-only field)</td>
<td>The logical name for the database.</td>
</tr>
<tr>
<td>Component model database path</td>
<td>The complete path name for the database .db file.</td>
</tr>
<tr>
<td>Service/port for server (-S)</td>
<td>The name or port of the database server.</td>
</tr>
<tr>
<td>Additional server parameters</td>
<td>Currently defined database startup parameters.</td>
</tr>
<tr>
<td>Automatically start/stop server</td>
<td>An option to have the database server start automatically when you start Progress Developer Studio for OpenEdge and stop when you quit.</td>
</tr>
</tbody>
</table>

**Connection information tab**

The following fields are available:

<table>
<thead>
<tr>
<th>Fields</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>The name of the remote database host machine, or localhost for a local database.</td>
</tr>
<tr>
<td>Service/Port</td>
<td>The name or port for the database server.</td>
</tr>
<tr>
<td>Database</td>
<td>The logical name of the database.</td>
</tr>
<tr>
<td>User</td>
<td>A valid database user name.</td>
</tr>
<tr>
<td>Password</td>
<td>The password for the specified user, if required.</td>
</tr>
</tbody>
</table>

**Projects tab**

The table lists the projects in your workspace. Projects with a check mark are associated with the model database, meaning that components created in those projects are stored in the database.

**Command buttons**

The following commands are available on all tabs:
**Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restore Defaults</td>
<td>To discard changes to database settings and revert to default values.</td>
</tr>
<tr>
<td>Apply</td>
<td>To save the current database settings and keep the Preferences dialog open.</td>
</tr>
<tr>
<td>OK</td>
<td>To save all current values and close the Preferences dialog.</td>
</tr>
<tr>
<td>Cancel</td>
<td>To close the Preferences dialog without saving any changes.</td>
</tr>
</tbody>
</table>

**See also**

- Component models on page 855
- Model storage options on page 856
- Creating a model database on page 858

### Create Component Model Database wizard

The Create Component Model Database wizard lets you create a database to store your component model. To start the wizard, select **File > New > Other > OpenEdge > Tools for Business Logic > Create Component Model Database** and click **Next**.

**Database name**

The **Component Database Name** dialog is the first step in creating a new model component database. You open this dialog by selecting **File > New > Other > OpenEdge > Tools for Business Logic > Create Component Model Database**. (You can also start the Create Component Model Database wizard by clicking **New** at the Component Model Database page in the Tools for Business Logic preferences section.)

The following fields and commands are available:

<table>
<thead>
<tr>
<th>Field</th>
<th>Component database name</th>
<th>A logical name for the database. This need not be the same as the name of the .db file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commands</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Next</td>
<td>To proceed to specify a path, server, and other parameters for the database.</td>
</tr>
<tr>
<td></td>
<td>Cancel</td>
<td>To close the Create Component Model Database wizard without creating a database.</td>
</tr>
</tbody>
</table>

**See also**

- Application models on page 855
- Model storage options on page 856
- Creating a model database on page 858
- Component Designer options on page 873
Local database path and server

Specifying a database file and server parameters is the second step in creating a new model component database. You open this dialog by clicking Next after entering a name for the database.

The following fields and commands are available:

<table>
<thead>
<tr>
<th>Fields</th>
<th>Create new database</th>
<th>A flag to make the other fields writable. If creating a local database, click this checkbox; if using a remote database, leave the field unchecked.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Component model database path</td>
<td>The complete path name for the database .db file.</td>
</tr>
<tr>
<td></td>
<td>Service/port for server (-S)</td>
<td>The name or port to use for the database server.</td>
</tr>
<tr>
<td></td>
<td>Additional server parameters</td>
<td>Any valid database startup parameters.</td>
</tr>
<tr>
<td></td>
<td>Automatically start/stop server</td>
<td>An option to have the database server start automatically when you start Progress Developer Studio for OpenEdge and stop when you quit.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Commands</th>
<th>Back</th>
<th>To return to the preceding page of the Create Component Model Database wizard.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Next</td>
<td>To proceed to specify database connection parameters.</td>
</tr>
<tr>
<td></td>
<td>Finish</td>
<td>To create the local database with default connection parameters and without associating projects.</td>
</tr>
<tr>
<td></td>
<td>Cancel</td>
<td>To close the Create Component Model Database wizard without creating a database.</td>
</tr>
</tbody>
</table>

See also
Application models on page 855
Model storage options on page 856
Creating a model database on page 858
Component Designer options on page 873

Database connection parameters

Database connection parameters are required for a remote database. Entering or changing values on this page is optional when you create a local component model database. You open this dialog by clicking Next at the dialog for specifying a local database path.

The following fields and commands are available:
Fields | Host | The name of the remote database host machine, or localhost for a local database. |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Service/Port</td>
<td>The name or port for the database server.</td>
</tr>
<tr>
<td></td>
<td>Database</td>
<td>The logical name of the database.</td>
</tr>
<tr>
<td></td>
<td>User</td>
<td>A valid database user name. For a local database, your login name is used by default.</td>
</tr>
<tr>
<td></td>
<td>Password</td>
<td>The password for the specified user, if required.</td>
</tr>
</tbody>
</table>

Commands | Back | To return to the preceding page of the Create Component Model Database wizard. |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Next</td>
<td>To proceed to associate projects with the model database.</td>
</tr>
<tr>
<td></td>
<td>Cancel</td>
<td>To close the Create Component Model Database wizard without saving any changes.</td>
</tr>
</tbody>
</table>

See also
- Application models on page 855
- Model storage options on page 856
- Creating a model database on page 858
- Component Designer options on page 873

Project association

You must explicitly associate with your model database those projects in which you intend to create Tools for Business Logic components. You can do so at the last page of the Create Component Model Database wizard. This dialog opens when you click Next at the dialog for specifying database connection parameters.

The following fields and commands are available:

Fields | Available Projects | The list of your existing projects in the current workspace. |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Selected Projects</td>
<td>The list of projects that you have selected to associate with the model database.</td>
</tr>
<tr>
<td>Commands</td>
<td>Add</td>
<td>To move the projects selected in the Available Projects field to the Selected Projects field.</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Remove</td>
<td>To move the projects selected in the Selected Projects field to the Available Projects field.</td>
</tr>
<tr>
<td></td>
<td>Remove All</td>
<td>To move all projects in the Selected Projects field to the Available Projects field.</td>
</tr>
<tr>
<td></td>
<td>Add All</td>
<td>To move all projects in the Available Projects field to the Selected Projects field.</td>
</tr>
<tr>
<td></td>
<td>Back</td>
<td>To return to the preceding page of the Create Component Model Database wizard.</td>
</tr>
<tr>
<td></td>
<td>Finish</td>
<td>To complete the creation of the model database.</td>
</tr>
<tr>
<td></td>
<td>Cancel</td>
<td>To close the Create Component Model Database wizard without creating a database.</td>
</tr>
</tbody>
</table>

### See also
- Application models on page 855
- Model storage options on page 856
- Creating a model database on page 858
- Component Designer options on page 873

### New Business Logic Component wizard

The New Business Logic Component wizard lets you create a new component in the Component Designer. To start the wizard, select File > New > Other > OpenEdge > Tools for Business Logic > Business Logic Component and click Next.

### New component name and location

The first page of the New Business Logic Component wizard lets you specify a name for the component, the component type, and a location within your project.

The following fields and commands are available:

<table>
<thead>
<tr>
<th>Fields</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>The folder in which the resource is to reside; this must be the project root folder or one of its subfolders.</td>
</tr>
<tr>
<td>Component name</td>
<td>The file name of the diagram (.dgm) file</td>
</tr>
<tr>
<td>Select a Component Type</td>
<td>Radio buttons for specifying the type of Business Logic component to create (Temp table or ProDataSet).</td>
</tr>
</tbody>
</table>
### Commands

<table>
<thead>
<tr>
<th>Back</th>
<th>To return to the wizard selection menu.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next</td>
<td>(Available only when XMI storage is in use and advanced wizard features are enabled) To proceed to select a model file for the component.</td>
</tr>
<tr>
<td>Finish</td>
<td>(Available only when advanced wizard features are disabled or when DB storage is in use) To create the specified component diagram and associate it with the default model file, appModel.t4bl, or with your component model database.</td>
</tr>
<tr>
<td>Cancel</td>
<td>To close the wizard without creating a new component.</td>
</tr>
</tbody>
</table>

### See also
- Component types on page 853
- Design output on page 855
- Creating a new component on page 861
- Determining how models are stored on page 858
- Creating a model database on page 858
- Component Designer options on page 873

### Model file selection

If you use the XMI model storage option and you have not disabled advanced wizard features, you must specify a new or existing .t4bl file for each component that you create. You open this dialog by clicking **Next** after specifying the component name, type, and location.

The following field and commands are available:

<table>
<thead>
<tr>
<th>Field</th>
<th>T4BL model files</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A list of available .t4bl files from which you select one to associate with the new component you are creating. This list is not populated until you complete a Create or Browse command. If there are multiple .t4bl files listed on completion of this dialog, the first model in the list is the one to which the component will be added. Subsequent models in the list will also appear in the Outline view, allowing you to reuse their components in the new diagram that you are creating.</td>
</tr>
<tr>
<td>Commands</td>
<td>Create</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Browse Workspace</td>
</tr>
<tr>
<td></td>
<td>Browse File System</td>
</tr>
<tr>
<td></td>
<td>Remove</td>
</tr>
<tr>
<td></td>
<td>Back</td>
</tr>
<tr>
<td></td>
<td>Finish</td>
</tr>
<tr>
<td></td>
<td>Cancel</td>
</tr>
</tbody>
</table>

**See also**

- Component types on page 853
- Design output on page 855
- Creating a new component on page 861
- Component models on page 855
- Determining how models are stored on page 858
- Component Designer options on page 873
Introducing Progress Developer Studio for OpenEdge customization

You can customize the following aspects of your Progress Developer Studio for OpenEdge environment:

- **Menu and toolbar options** - You can add commands that run ABL procedures (typically custom tool applications) to a menu or the toolbar in the OpenEdge Editor perspective, or to the context (right-click) menu of the ABL Editor.

- **Code-generation templates** - You can add custom templates, or override those that are installed with Progress Developer Studio for OpenEdge, to define menu options and the output generated when those options are used.

- **Startup routines** - You can define ABL procedures that run automatically when Progress Developer Studio for OpenEdge starts.

- **Event subscription** - Progress Developer Studio for OpenEdge publishes events when various operations, such as opening or saving a file, occur. You can write ABL procedures that subscribe to and respond to these events.

- **Custom project types** - You can create project types with customized folder layouts and PROPATH settings.

**Note:** OpenEdge stores the customized menus, toolbars, templates, and project types in the com.openedge.pdt.extensibility.user Eclipse plugin. You can specify a different plugin for this purpose, if necessary, on the [Customization preferences page](#).

For details, see the following topics:

- **Concepts**
Concepts

Menu and toolbar customization

You can customize your environment so that any executable ABL procedure can be launched from:

- An OpenEdge Editor perspective menu that is reserved for user-defined commands. You determine the name of this menu.
- A submenu of the context menu that appears when you right-click in an ABL Editor window.
- A button on the OpenEdge Editor perspective toolbar.

To change menus or the toolbar in these ways, you use the Customization Editor.

See also
- Introducing Progress Developer Studio for OpenEdge customization on page 883
- Using the Customization Editor on page 886
- Customization Editor on page 896

Template customization

Templates are the basis for new source code elements. When you use the File > New menu to create a new resource, or when you right-click in the ABL Editor and use the Source submenu, an underlying template determines the content of the file that is generated.

Progress Developer Studio for OpenEdge provides standard templates for the files and code segments available from these menus. You can create your own templates and use the Customization Editor to change the installed options as follows:

- You can add options to the File > New menu. For each template that you add, Progress Developer Studio for OpenEdge automatically creates a menu entry and a corresponding wizard. When the entry is chosen, the wizard generates a new file containing the custom code.
- You can override the standard templates used by the wizards launched from the File > New menu or the Source submenu. The wizards then generate code based on your custom templates instead of the standard ones.
- You can associate the templates to custom project types.

Note: You must have access to the JET templates that you use and they must be in your Java class path. On the Customization preferences page, you can easily add template directories to your class path.

Information resources for JET templates

The following are some useful sources of information about templates:
Startup customization

You can write an ABL procedure to run automatically when you start Progress Developer Studio for OpenEdge. For example, you might create a startup procedure that subscribes to Progress Developer Studio for OpenEdge events.

See also
Customizing Project Startup on page 892

Event subscription

Progress Developer Studio for OpenEdge publishes a named event each time any of several system operations (for example, opening or closing a file) occurs. By using the SUBSCRIBE statement in ABL code, you can capture these events and respond to them programmatically.

See also
Subscribing to OpenEdge Architect Events on page 892

Custom project types

Progress Developer Studio for OpenEdge allows you to create a custom project type which is a combination of different project types added to a single project. In a custom project type, you can specify the folder layout, the default files in each folder, and the PROPATH settings for folders in each project. You can also share these custom project types with other developers. The custom project types you create can be used as templates for new projects.

For more information on the options for creating a custom project type, see Customization Editor on page 896.

Note: OpenEdge stores the details of the custom project types in the com.openedge.pdt.extensibility.user Eclipse plugin. So, if you are running Progress Developer Studio for OpenEdge in multiple Eclipse environments, you must ensure that the required projects are available for each Eclipse environment.

See also
Using the Customization Editor on page 886
Customization Editor on page 896
Tasks

Setting customization preferences

To open the Customization preferences page, select Window > Preferences > Progress > OpenEdge > Advanced > Customization. On this page, you can:

- Specify the Eclipse plugin containing the custom menu and toolbar definitions to be used. Generally, there is no need to change the default.
- Identify the locations in which you store custom templates.

Customization plugin

By default, menu, toolbar, template, and project type customizations are stored in a plugin named com.openedge.pdt.extensibility.user. It is possible to create and maintain additional plugins with different customization definitions. If you have multiple plugins, specify in the Customization plugin ld field the one in which you want to store changes that you make with the Customization Editor. The value in this field must correspond to a valid Eclipse plugin.

If multiple customization plugins are present, specifying one on the Customization preferences page does not disable the others. It just identifies the plugin in which subsequent changes are saved. The potential exists for conflicts in settings. To avoid problems, consider making only one customization plugin available to Eclipse at any given time.

Template directories

The location of any custom template you want to use must be in your Java class path. Selecting All open project directories, entering a specific location in the Template folder field, or both causes the specified directories to be added automatically to your class path. You cannot specify multiple directories in the Template folder field.

See also

Menu and toolbar customization on page 884
Template customization on page 884
Customization Preferences dialog on page 904

Using the Customization Editor

Adding menu and toolbar commands

You use the Customization Editor to add ABL programs to the OpenEdge Editor perspective menu bar or toolbar, or to the ABL Editor context (right-click) menu. After adding options, you can also use this tool to edit or remove them.

Follow these steps to add a program to the menus or toolbar:
1. In the OpenEdge Editor perspective, open the Customization Editor by selecting OpenEdge > Tools > Customization Editor.

2. In the Available Extension Points tree view, click either Menu/Toolbar Entries or Editor Context Menu.
   If you select Menu/Toolbar Entries, you can change the menu label, which by default is Extensibility. This is the label that will appear on the menu bar. To designate a character in the label as a shortcut key in combination with the Alt key, enter an ampersand (&) before that character. (For example, if you keep the default value, Extensibility, pressing Alt+Y opens the menu.)

3. Click Add. As a shortcut to skip steps 2 and 3, click or on the Customization Editor toolbar to add a menu/toolbar entry or a context menu entry, respectively.

4. Enter the appropriate values for the fields that appear under Action Details. See the Customization Editor reference help for detailed information about these fields.

5. If you want the program to run as a persistent procedure, check the Run persistent option at the bottom.

6. If you selected Menu/Toolbar Entries in step 2, click the appropriate radio button in the Action appearance section to indicate whether the option should be placed on the menu, the toolbar, or both.

7. Save your changes (Select File > Save or click )

8. To make the newly added option available immediately, click (rightmost on the Customization Editor toolbar) to reset the OpenEdge Editor perspective. Then, click Yes in the restart prompt that appears. Your changes do not take effect until you restart Progress Developer Studio for OpenEdge.

See also
Menu and Toolbar Customization on page 884
Editing and deleting options on page 891
Customization Editor on page 896

Customizing code templates
You can use the Customization Editor to add to the File > New menu, file-creation wizards that use custom Java Emitter Templates (JET templates) and Editor templates.
You must have access to custom JET templates for creating ABL code to be able to add file-creation wizards that use these templates.
You can also override the standard templates used by the options on that menu or the ABL Editor context (right-click) menu. After defining template customizations, you can use the Customization Editor to edit or remove them. You can also associate these templates to custom project types.
For information about creating JET templates, see OpenEdge Architect: Working with Custom Templates in the Product Documentation section of the Progress Software Developers Network Web site.
Your template files must be in your Java class path. You can specify template directories to be added to your class path in Customization Preferences.
Begin by opening the Customization Editor. In the OpenEdge Editor perspective, select OpenEdge > Tools > Customization Editor.

1. Follow these steps to add a new JET template entry to the File > New menu:
   a) In the Available Extension Points tree view, click New JET Templates.
   b) Click Add.
   c) Enter a name, and optionally an image file for an accompanying icon, to appear on the File > New menu.
d) In the **Template** file field, enter the name of the Java class file containing the compiled JET template.

e) **Save** your changes.

2. Follow these steps to add a new Editor template entry to the **File > New** menu:

   a) In the **Available Extension Points** tree view, click **Editor Templates**.

   b) Click **New**.

   c) Enter the appropriate values for the fields that appear in the New Template dialog. See **Editor Templates options** on page 899 for detailed information about these fields.

   d) **Save** your changes.

   e) You can also edit, remove, import, and export Editor templates. See **Editor Templates options** on page 899 for detailed information about these operations.

3. Follow these steps to specify a template for use in place of a standard Progress Developer Studio for OpenEdge template:

   a) In the **Available Extension Points** tree view, click **Template Overrides**.

   b) Click **Add**.

   c) In the **Template** field, select the standard template that you want to override from the drop-down list.

   d) In the **Template file** field, enter the name of the Java class file containing the compiled JET template to be used instead.

   e) **Save** your changes.

4. After adding or overriding template definitions:

   a) Save your changes by selecting **File > Save** or clicking ![Save icon](save_icon.png).

   b) To make the changes take effect, click ![Reset icon](reset_icon.png) (rightmost on the Customization Editor toolbar) to reset the OpenEdge Editor perspective. Then, click **Yes** in the restart prompt that appears. Your changes do not take effect until you restart Progress Developer Studio for OpenEdge.

**See also**

- **Template customization** on page 884
- **Editing and deleting options** on page 891
- **Setting customization preferences** on page 886
- **Customization Editor** on page 896

**Working with custom project types**

You can use the Customization Editor to create, edit, remove, import, and export custom project types.

**See also**

- **Custom Project Types** on page 885
- **Customization Editor** on page 896
Creating custom project types

You can use the Customization Editor to create custom project types. Once you create the custom project types, you can use this tool to edit, remove, import, and export them. The custom project types that you create appear on the Create an OpenEdge Project page of the New OpenEdge Project wizard.

Follow these steps to create a custom project type:

1. In the OpenEdge Editor perspective, open the Customization Editor by selecting OpenEdge > Tools > Customization Editor.
2. In the Available Extension Points tree view, click OpenEdge Custom Projects.
3. Click Add.

**Note:** Optionally, you can click on the Customization Editor toolbar to create a custom project type.

4. Enter the appropriate values for the fields that appear under Project Type Configuration. See the Customization Editor reference help for detailed information about these fields.

5. Save your changes (Select File > Save or click .)
6. Restart Progress Developer Studio for OpenEdge to make your changes take effect.

**Note:** The Customization Editor displays a warning message each time you make changes to a custom project.

See also

Custom Project Types on page 885
Editing and removing Custom Project Types on page 889
Importing and Exporting Custom Project Types on page 890
Customization Editor on page 896

Editing and removing custom project types

Once you create custom project types using the Customization Editor, you can use the same tool to edit or remove them. To do so, follow these steps:

1. In the OpenEdge Editor perspective, open the Customization Editor by selecting OpenEdge > Tools > Customization Editor.
2. In the Available Extension Points tree view, expand the OpenEdge Custom Projects node by clicking the plus symbol (+), and select the entry you want to change.
3. Make the required changes in the Project Type Configuration section, or click Remove to delete the entry.
Note: To remove a custom project type, you must uninstall it from all its associated OpenEdge projects that are currently open in the workspace. Otherwise, an information dialog appears with the names of all the associated OpenEdge projects and prompts you to uninstall the associations. This dialog appears only for the projects open in the current workspace. If there are multiple workspaces using the same Progress Developer Studio for OpenEdge, you must ensure that the custom project type is removed from all projects in all the applicable workspaces.

4. Save your changes (Select File > Save or click ![save icon]).

5. Restart Progress Developer Studio for OpenEdge to make your changes take effect. When you open or import an OpenEdge project associated with a project type that has been removed or not yet configured in the current workspace, a warning message appears in the Project Facets project properties page.

To remove the project type association with the OpenEdge project, you must remove the custom project entry from the org.eclipse.wst.common.project.facet.core.xml file located in the .settings folder of the project.

Note: Removing this entry from the XML file does not remove the artifacts created by the project type. You must remove them manually.

See also
Custom Project Types on page 885
Creating Custom project Types on page 889
Importing and Exporting Custom Project Types on page 890
Customization Editor on page 896

Importing and exporting custom project types

OpenEdge stores the details of custom project types that you create in the com.openedge.pdt.extensibility.user Eclipse plugin (you can specify a different plugin, if necessary, on the Customization preferences page). The details of custom project types created by other users are stored in similar user plugins. You can import these custom project types using the Customization Editor. You can also export the custom project types that you create.

1. To import custom project types:
a) In the OpenEdge Editor perspective, open the Customization Editor by selecting **OpenEdge > Tools > Customization Editor**.

b) In the **Available Extension Points** tree view, click OpenEdge Custom Projects.

   a) Click **Import**.

   a) Select a plugin folder of another user from the list of directories. The **Import Custom Project Types** dialog appears with a list of custom projects available in that plugin folder.

   a) Select the custom project types you want to import.

   **Note:** The dialog displays an error if you select a project type that already exists in the workspace.

2. To export custom project types:

   a) In the OpenEdge Editor perspective, open the Customization Editor by selecting **OpenEdge > Tools > Customization Editor**.

   a) In the **Available Extension Points** tree view, click **OpenEdge Custom Projects**.

   b) Select the project types that you want to export on the Export Custom Project Types dialog.

   a) Click **Export**. The dialog creates a user plugin with the details of the selected custom project types that can be used by other users.

**See also**

Custom Project Types on page 885  
Creating Custom Project Types on page 889  
Editing and Removing Custom Projects on page 889  
Customization Editor on page 896

**Editing and deleting options**

After adding custom options with the Customization Editor, you can use the same tool to edit or remove them. Follow these steps:

1. In the OpenEdge Editor perspective, open the Customization Editor by selecting **OpenEdge > Tools > Customization Editor**.

2. In the **Available Extension Points** tree view, expand the appropriate node by clicking the plus symbol (+), and select the entry you want to change.

3. Make the desired changes in the Details section, or click **Remove** to delete the entry.

4. Save your changes (Select **File > Save** or click ![Save](button.png)).

5. To make the changes effective, click ![Reset](button.png) (rightmost on the Customization Editor toolbar) to reset the OpenEdge Editor perspective. Then, click **Yes** in the restart prompt that appears. Your changes do not take effect until you restart Progress Developer Studio for OpenEdge.

**See also**

Adding menu and toolbar commands on page 886  
Customizing code templates on page 887  
Customization Editor on page 896
Sharing customization settings

To share custom menu, toolbar, and template settings with another user, do the following:

1. You must provide to that user:
   a) An archive such as a .ZIP file containing your customization plugin directory and all of its contents. The default customization plugin is:
   
   OpenEdge-install-dir\OpenEdge\oeide\eclipse\plugins\com.openedge.pdt.extensibility.user_version_number
   
   b) Copies of all custom templates.

2. The other user should then do the following:
   a) Extract the contents of the plugin archive to the same location on her or his own machine, and confirm that the value of the Version field in the Customization Editor matches the version number in the directory name.
   b) Save the custom templates in the desired location, and make sure that location is in the Java class path.

See also

Menu and toolbar customization on page 884
Template customization on page 884
Customization Preferences page on page 904

Customizing project startup

When you open a Progress Developer Studio for OpenEdge project, the OpenEdge Runtime searches the project's PROPATH for a file named _idestartup.p. If that file is found, any procedures that it contains are automatically executed. Therefore, you can implement a custom startup routine by creating the _idestartup.p. procedure.

For example, you might write a startup procedure that subscribes to Progress Developer Studio for OpenEdge events.

Reference

Subscribing to OpenEdge Architect events

Each time a Progress Developer Studio for OpenEdge operations takes place, a named event, oeide_event, is published. This enables you to develop procedures that use the ABL SUBSCRIBE statement to capture and respond to Progress Developer Studio for OpenEdge events.

Output parameters for oeide_event

The following output parameters are published:
<table>
<thead>
<tr>
<th>Event Name</th>
<th>Output Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventName</td>
<td>One of the following values:</td>
</tr>
<tr>
<td></td>
<td>• Open-file</td>
</tr>
<tr>
<td></td>
<td>• Close-file</td>
</tr>
<tr>
<td></td>
<td>• Before-save</td>
</tr>
<tr>
<td></td>
<td>• After-save</td>
</tr>
<tr>
<td></td>
<td>• Before-compile</td>
</tr>
<tr>
<td></td>
<td>• After-compile</td>
</tr>
<tr>
<td></td>
<td>• Before-run</td>
</tr>
<tr>
<td></td>
<td>• After-run</td>
</tr>
<tr>
<td></td>
<td>• Before-debug</td>
</tr>
<tr>
<td></td>
<td>• After-debug</td>
</tr>
<tr>
<td></td>
<td>• Project-startup</td>
</tr>
<tr>
<td></td>
<td>• Project-shutdown</td>
</tr>
<tr>
<td></td>
<td>• Get-focus</td>
</tr>
<tr>
<td>projectName</td>
<td>The name of the project in which the event occurs.</td>
</tr>
<tr>
<td>programName</td>
<td>The file name of the resource operated on, if applicable.</td>
</tr>
<tr>
<td>eventData</td>
<td>(Not currently used.)</td>
</tr>
</tbody>
</table>

**Note:** In certain circumstances, event publishing is suppressed so as not to interfere with a running process. For example, opening or closing a file while running the Progress Developer Studio for OpenEdge Debugger does not trigger publication of an event. Also, note that the Before-compile and After-compile notifications do not apply to Tools for Business Logic, as models are only saved and not compiled.

**AppBuilder events**

When AppBuilder runs in Progress Developer Studio for OpenEdge, most of its events are published by both OpenEdge Studio and Progress Developer Studio for OpenEdge. Whenever possible, it is recommended that you use the Progress Developer Studio for OpenEdge events, rather than the corresponding OpenEdge Studio events. However, the following OpenEdge studio events do not have Progress Developer Studio for OpenEdge counterparts:

- New
- Before-Open
- Before-Close
- Before-Check-Syntax
- Check-Syntax
- Before-Check-Syntax-Partial
- Check-Syntax-Partial
Note: The AppBuilder in Progress Developer Studio for OpenEdge does not have a Compile option. If the ABL Editor is opened for an AppBuilder file, then the compile notifications will be sent only if the file is saved under OpenEdge Projects.

Named event publishing for AppServers in Progress Developer Studio for OpenEdge

Progress Developer Studio for OpenEdge currently publishes events on the AVM for certain events, such as starting a project, stopping a project, and compiling.

The following events are new in this release:

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>before-server-publish</td>
<td>Sent to all OpenEdge projects that contribute one or more modules to a server. This event is published when the server starts to publish.</td>
</tr>
<tr>
<td>after-server-publish</td>
<td>Sent to all OpenEdge projects that contribute one or more modules to a server. This event is published when the server finishes publishing.</td>
</tr>
<tr>
<td>before-module-publish</td>
<td>Sent to all OpenEdge projects that contribute one or more modules to a server. This event is published when the server begins to publish a specific module.</td>
</tr>
<tr>
<td>after-module-publish</td>
<td>Sent to all OpenEdge projects that contribute a specific module to a server. This event is published after the server publishes a specific module.</td>
</tr>
<tr>
<td>server-starting</td>
<td>Sent when an OpenEdge server is known to be starting (typically when the launch for a particular server has started). This event is published only to projects that contribute one or more modules to the server.</td>
</tr>
<tr>
<td>server-started</td>
<td>Sent when an OpenEdge server is known to have started from within a Progress Developer Studio for OpenEdge launch configuration. This event is published from both the launch configuration (after it detects that the server has started) and from the server process that monitors the server state. This event is published only to projects that contribute one or more modules to the server.</td>
</tr>
</tbody>
</table>
Event Name | Description
---|---
server-stopping | Sent when an OpenEdge server is stopping. This event is published:
  • Only to projects that contribute one or more modules to the stopped server
  • Whenever the framework detects that the server is stopping

server-stopped | Sent when an OpenEdge server is stopped. This event is published:
  • Only to projects that contribute one or more modules to the server
  • Whenever the framework detects that the server has stopped
  This event is called from both the stop process job and the server process that monitors the state of the server.

Each event is published in the AVM with four parameters. The following table shows the value of each parameter for each event:

<table>
<thead>
<tr>
<th>Event</th>
<th>Project name</th>
<th>Program name</th>
<th>Event data</th>
</tr>
</thead>
<tbody>
<tr>
<td>before-server-publish</td>
<td>Project name of module owner</td>
<td>&lt;blank&gt;</td>
<td>Server name</td>
</tr>
<tr>
<td>after-server-publish</td>
<td>Project name of module owner</td>
<td>&lt;blank&gt;</td>
<td>Server name</td>
</tr>
<tr>
<td>before-module-publish</td>
<td>Project name of module owner</td>
<td>Module name</td>
<td>Server name</td>
</tr>
<tr>
<td>after-module-publish</td>
<td>Project name of module owner</td>
<td>Module name</td>
<td>Server name</td>
</tr>
<tr>
<td>server-starting</td>
<td>Project name of module owner</td>
<td>&lt;blank&gt;</td>
<td>Server name</td>
</tr>
<tr>
<td>server-started</td>
<td>Project name of module owner</td>
<td>&lt;blank&gt;</td>
<td>Server name</td>
</tr>
<tr>
<td>server-stopping</td>
<td>Project name of module owner</td>
<td>&lt;blank&gt;</td>
<td>Server name</td>
</tr>
<tr>
<td>server-stopped</td>
<td>Project name of module owner</td>
<td>&lt;blank&gt;</td>
<td>Server name</td>
</tr>
</tbody>
</table>

**Sample event-subscription startup routine**

The following example shows a startup file, _idestartup.p, that subscribes to Progress Developer Studio for OpenEdge events and runs event_alert.p as a persistent procedure:
Chapter 17: Introducing Progress Developer Studio for OpenEdge customization

Customization Editor

You use the Customization Editor to:

- Define custom entries on the OpenEdge Editor perspective menu and toolbar.
- Define custom entries on the ABL Editor context (right-click) menu.
- Associate custom JET templates with new code-generation wizards to be added to the File > New menu.
- Associate standard code-generation wizards (those installed with Progress Developer Studio for OpenEdge) with custom JET templates, overriding their association with standard installed templates.
- Create, edit, remove, import, and export custom project types.

You open the Customization Editor in the OpenEdge Editor perspective by selecting OpenEdge > Tools > Customization Editor.

The controls available in the Editor depend on the selection context - that is, the current selection in the Available Extension Points tree view.

See also
Menu and toolbar customization on page 884
Template customization on page 884
Custom Project Types on page 885
## Controls available in all contexts

<table>
<thead>
<tr>
<th>Controls</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available Extension Points</td>
<td>A tree view in which you select the type of customization entry to add or an existing entry to edit.</td>
</tr>
<tr>
<td>Add Program to Menu/Toolbar</td>
<td>Displays the form for a new entry under the Menu/Toolbar Entries node.</td>
</tr>
<tr>
<td>Add Program to Editor Menu</td>
<td>Displays the form for a new entry under the Editor Context Menu node.</td>
</tr>
<tr>
<td>Add Custom Template</td>
<td>Displays the form for a new entry under the New Templates node.</td>
</tr>
<tr>
<td>Add Template Override</td>
<td>Displays the form for a new entry under the Template Overrides node.</td>
</tr>
<tr>
<td>Add Custom Project</td>
<td>Displays the form for a new entry under the OpenEdge Custom Projects node.</td>
</tr>
<tr>
<td>Refresh Customization Options</td>
<td>Resets the OpenEdge Editor perspective after editing and prompts you to restart Progress Developer Studio for OpenEdge for the changes to take effect. You are first prompted to save changes if necessary.</td>
</tr>
<tr>
<td>Add</td>
<td>Displays the form for a new entry of the type selected in the tree view.</td>
</tr>
<tr>
<td>Remove</td>
<td>Deletes the currently selected entry.</td>
</tr>
<tr>
<td>Version</td>
<td>A string that must match the version extension of the customization plugin name. This field is disabled if a non-default plugin (that is, any other than com.openedge.pdt.extensibility.user) is in use.</td>
</tr>
<tr>
<td></td>
<td>If you change the version number here, a copy of com.openedge.pdt.extensibility.user is created with this version number.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>There is no need to change this value, and you should not do so, unless you are very familiar with Eclipse plugin management and are using advanced plugin configuration techniques.</td>
</tr>
</tbody>
</table>

### See also

- [Menu and toolbar customization](#) on page 884
- [Template customization](#) on page 884
- [Custom project types](#) on page 885
- [Using the Customization Editor](#) on page 886
# Menu/Toolbar Entries or Editor Context Menu

The following controls are available for this extension point:

<table>
<thead>
<tr>
<th>Controls</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Menu label</strong></td>
<td>(Available only when top-level Menu/Toolbar Entries node is selected) The text to appear on the menu bar as the name of the custom menu. Optionally, an ampersand (&amp;) before a character in this string makes that character a shortcut key (pressing Alt plus the shortcut key opens the menu).</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td>The name of the option. If you are adding a menu option, this is the text that will appear on the menu.</td>
</tr>
</tbody>
</table>
| **Accelerator**| (Optional) A key sequence that can be used to activate the program without use of the mouse. You can use Ctrl, Alt, or both, plus a shifted or unshifted character or function key. You must type the key names, separated with + (no spaces). The following examples are valid accelerator sequences:  
  - Ctrl+Shift+Z  
  - Alt+S  
  - Ctrl+Alt+Shift+F10  
  If you are adding a menu option, the assigned key sequence will appear on the menu along with the program name.  
  The sequence must not be assigned to another function. If you specify an accelerator that is already in use, it will be disabled for both operations. To view existing assignments, select **Window > Preferences > General > Keys**. |
| **Icon**      | (Optional) An image to appear on the menu and/or toolbar button. The image file must be in the icons directory of the extensibility plugin (by default, com.openedge.pdt.extensibility.user), or a subdirectory in that path.  
  An image size of 16x16 pixels is recommended for best results.  
  If you add a toolbar option without specifying an icon, the program name appears on the toolbar. |
| **Tooltip**   | (Optional) Text to appear in a popup window when the user rests the mouse pointer over the toolbar button. If you leave this field blank, the option name is used.  
  Tooltips have no effect in menus. |
| **Program name**| The executable ABL program (.p, .w, or .r file) to run when the menu or toolbar option is chosen. If the file is not in your Propath, include the full path.  
  The value of this field works in conjunction with the value of the Configuration field, as follows:  
  - If you want this program to run externally, also choose a launch configuration in the Configuration field. The program you enter overrides the initial program specified in the configuration definition.  
  - If you simply want the menu or toolbar option to launch an external configuration and run the initial program as specified in the configuration definition, choose the appropriate entry in the Configuration field and leave the Program name field blank. |
### Controls

<table>
<thead>
<tr>
<th>Parameters</th>
<th>(Optional) The parameters, if any, with which the program is to run.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration</td>
<td>The drop-down list for this field shows the launch configurations defined for your workspace. Choose the launch configuration under which you want the menu or toolbar option to run the program. (See the instructions above for the Program name field.)</td>
</tr>
<tr>
<td>Action appearance</td>
<td>(Available only when adding an entry under Menu/Toolbar Entries) A radio button set that lets you specify whether the option is to appear on the menu, the toolbar, or both.</td>
</tr>
<tr>
<td>Run persistent</td>
<td>An option to run the specified program as a persistent procedure.</td>
</tr>
</tbody>
</table>

#### See also
- Menu and toolbar customization on page 884
- Using the Customization Editor on page 886

### New JET Templates options

The following controls are available for this extension point:

<table>
<thead>
<tr>
<th>Controls</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the wizard. This text will appear on the File &gt; New menu.</td>
</tr>
<tr>
<td>Icon</td>
<td>(Optional) An image to appear on the menu. The image file must be in the icons directory of the extensibility plugin (by default, com.openedge.pdt.extensibility.user), or a subdirectory in that path. An image size of 16x16 pixels is recommended for best results.</td>
</tr>
<tr>
<td>Template file</td>
<td>The name of the Java class file containing the template that the new wizard is to use for code generation. The file must be in your Java class path.</td>
</tr>
</tbody>
</table>

#### See also
- Template customization on page 884
- Using the Customization Editor on page 886

### Editor Templates options

The Customization Editor allows you to create, edit, and remove Editor templates. You can associate these templates to the custom project types using the Code-generation Templates option. Associating an Editor template to a custom project type creates a New File option in the File > New menu, which you can use to create a new file with the selected template pattern.

When you select the Editor Templates node, the Customization Editor displays the New Editor Templates section in its right pane. In this section, you can create, edit, remove, import, and export Editor templates.

The following controls are available in the New Editor Templates section:
Open the New Template dialog that allows you to create a new Editor template.

Allows you to edit the currently selected Editor template options in the New Template dialog and change the name of the template. When you change the name of the template, an Edit Template dialog appears, asking if you want to add a new template with that name or modify an existing one.

Deletes the currently selected entry.

Allows you to import Editor templates from the file system.

Allows you to export Editor templates to the file system.

<table>
<thead>
<tr>
<th>Controls</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>New</td>
<td>Opens the New Template dialog that allows you to create a new Editor template.</td>
</tr>
<tr>
<td>Edit</td>
<td>Allows you to edit the currently selected Editor template options in the New Template dialog and change the name of the template. When you change the name of the template, an Edit Template dialog appears, asking if you want to add a new template with that name or modify an existing one.</td>
</tr>
<tr>
<td>Remove</td>
<td>Deletes the currently selected entry.</td>
</tr>
<tr>
<td>Import</td>
<td>Allows you to import Editor templates from the file system.</td>
</tr>
<tr>
<td>Export</td>
<td>Allows you to export Editor templates to the file system.</td>
</tr>
</tbody>
</table>

**Controls available for the New Template dialog**

<table>
<thead>
<tr>
<th>Controls</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Allows you to specify the name of the template.</td>
</tr>
<tr>
<td>Context</td>
<td>Determines where the template can be used and the set of available predefined template variables.</td>
</tr>
<tr>
<td>Description</td>
<td>Specifies the template description.</td>
</tr>
<tr>
<td>Pattern</td>
<td>Specifies the template pattern.</td>
</tr>
<tr>
<td>Insert Variables</td>
<td>Display the list predefined context specific variables.</td>
</tr>
</tbody>
</table>

**See also**

Template customization on page 884
Using the Customization Editor on page 886

**Template Overrides options**

The following controls are available for this extension point:

<table>
<thead>
<tr>
<th>Controls</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template</td>
<td>The name of the standard Progress Developer Studio for OpenEdge template to be overwritten.</td>
</tr>
<tr>
<td>Template file</td>
<td>The name of the Java class file containing the template to be used in place of the standard template. The file must be in your Java class path.</td>
</tr>
</tbody>
</table>

**See also**

Template customization on page 884
OpenEdge Custom Project options

The following controls are available for this extension point:

<table>
<thead>
<tr>
<th>Controls</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Creates a custom project type and displays the Project Type Configuration section in the right pane of the Customization Editor. See Controls available for Project Type Configuration. This option is also available on the Customization Editor toolbar ( ).</td>
</tr>
<tr>
<td><strong>Remove</strong></td>
<td>Deletes the currently selected entry.</td>
</tr>
<tr>
<td><strong>Import</strong></td>
<td>Allows you to import custom project types from other extensibility user plugins.</td>
</tr>
<tr>
<td><strong>Export</strong></td>
<td>Allows you to export custom project types that you create to other users.</td>
</tr>
</tbody>
</table>

**Controls available for Project Type Configuration**

You can use the Project Type Configuration section to configure the properties related to the custom project types that you create.

Following are the controls available for this section.

<table>
<thead>
<tr>
<th>Controls</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>Allows you to specify the name of the custom project type. Displays an error if you specify a name that already exists.</td>
</tr>
<tr>
<td><strong>Icon</strong></td>
<td>(Optional) Allows you to browse and select an image that appears on the Project Facets project properties page. The image file must be in the icons directory of the extensibility plugin (by default, com.openedge.pdt.extensibility.user), or a subdirectory in that path. An image size of 16x16 pixels is recommended for best results.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>(Optional) Allows you to add a brief description of the custom project type. This description is displayed on the Project Facets project properties page.</td>
</tr>
<tr>
<td><strong>Perspective</strong></td>
<td>(Optional) Allows you to choose from a list of available OpenEdge perspectives. This perspective is activated after creating an OpenEdge project that is associated with the current custom project type. The default perspective is OpenEdge Editor.</td>
</tr>
</tbody>
</table>
### Controls

<table>
<thead>
<tr>
<th>Required facet</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allows you to select a project type from a list of all the project types and custom project types (other than itself) in Progress Developer Studio for OpenEdge. The project type you select here is displayed on the Project Facets project properties page along with the default OpenEdge project type which is a required facet for every project type.</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** If the OpenEdge project type is not selected on the Project Facets project properties page for a custom project type, an error is displayed indicating the missing dependency.

<table>
<thead>
<tr>
<th>OpenEdge runtime</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allows you to select the runtime for the current project as either GUI or TTY. The default value is based on the selected project type.</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The GUI client is supported only on Windows. Therefore, on non-Windows platforms, TTY is automatically selected and cannot be changed. It does not change according to the type of project you select.

Each custom project type node contains the following nodes:

- **Folder Layout:** Allows you to specify the default folder structure of the project and the files in the folder. Also, allows you to add the folder to PROPATH. Clicking this node displays the Folder Layout section and the PROPATH section on the right pane of the Customization Editor.

- **Code-generation Templates:** Allows you to create a set of default templates and associate them to the custom projects. Associating the template to a custom project creates a New File option in the File > New menu, which you can use to create a new file with the selected template pattern.

**See also**

- Custom project Types on page 885
- Template Customization on page 884
- Using the Customization Editor on page 886
- Folder Layout Options on page 902
- Code Generation Template Options on page 904

### Folder Layout options

Selecting the Folder Layout node under a custom project type displays a Folder Layout section and a PROPATH section on the right pane of the Customization Editor.

Controls available for the Folder Layout section

The Folder Layout section displays the folders and files that you add in a tree format. The default parent node for all the folders and files is &lt;Project Root&gt;. The folders that can be added to PROPATH are displayed with a different icon.

**Note:** The modifications to the folder layout are saved by default as these operations (adding and removing) are done on file system directly.

The following controls are available for this section:
### Controls available for Add File/Folder and Edit File/Folder

The Add File/Folder dialog allows you to add files or folders to a custom project type. You can edit the file or folder options that you select on this dialog using the Edit File/Folder dialog (contains the same options).

The following controls are available on these dialogs:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Folder</strong></td>
<td>Enables options for adding a folder to the custom project type.</td>
</tr>
<tr>
<td><strong>File</strong></td>
<td>Enables options for adding a file to the custom project type.</td>
</tr>
<tr>
<td><strong>Folder name</strong></td>
<td>(Is enabled only when you select Folder) Specifies the name of the folder.</td>
</tr>
<tr>
<td><strong>Copy content from existing folder</strong></td>
<td>(Is enabled only when you select Folder) Allows you to load folder(s)/file(s) from the file system.</td>
</tr>
<tr>
<td><strong>Include sub-folders</strong></td>
<td>(Is enabled only when you select Copy content from existing folder) Loads content from all levels of the files/folders under the selected folder.</td>
</tr>
<tr>
<td><strong>Path</strong></td>
<td>(Common for Folder and File) Specifies the location of the folder or file in the file system.</td>
</tr>
<tr>
<td><strong>Add to PROPATH</strong></td>
<td>(Common for Folder and File) Adds the currently selected file or folder to PROPATH. The files or folders that you add here appear in the PROPATH section (on the right pane of the Customization Editor).</td>
</tr>
</tbody>
</table>

*Note:* The content under the specified folder is copied to the folder layout. The changes in the actual directory are reflected automatically.
Controls available for the PROPATH section

The PROPATH section displays the PROPATH entries that you add by using the Add to PROPATH option in the Folder Layout section. The default entry displayed in this section is @{ROOT} and it represents the project root folder. When you add a folder or file to PROPATH, that entry is added below @{ROOT}.

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move Up, Move Down</td>
<td>Changes the position of the current entry in the list and thus specifies the order of the PROPATH entries.</td>
</tr>
<tr>
<td>Remove</td>
<td>Deletes the currently selected entry and updates the corresponding folder or file image.</td>
</tr>
</tbody>
</table>

See also

Custom project Types on page 885
Template Customization on page 884
Using the Customization Editor on page 886
OpenEdge Custom Project Options on page 901

Code-generation Templates options

Selecting the Code-generation Templates node under a custom project type displays a list of the templates associated with the custom project type in the Templates section (appears in the right pane of the Customization Editor).

The following controls are available for this section:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Opens the Add Template Association dialog. This dialog allows you to specify the name of the template and choose the type of the template you want to associate to the custom project. The Add Template Association dialog provides two template options: New JET templates and Editor templates.</td>
</tr>
<tr>
<td>Edit</td>
<td>Allows you to edit the template options that you select in the Add Template Association dialog and change the name of the template.</td>
</tr>
<tr>
<td>Remove</td>
<td>Deletes the currently selected entry.</td>
</tr>
</tbody>
</table>

See also

Custom project Types on page 885
Template Customization on page 884
Using the Customization Editor on page 886
OpenEdge Custom Project Options on page 901

Customization Preferences page

The Customization Preferences page lets you:
- Specify the Eclipse plugin that is used for customizations.
- Automatically append template directories to your Java class path.

You open the Customization Preferences page by selecting **Window > Preference > Progress OpenEdge > Advanced > Customization.**

The following controls are available:

<table>
<thead>
<tr>
<th>Controls</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customization plugin Id</td>
<td>The name of the plugin in which menu and toolbar customizations are stored. The value in this field must correspond to a valid Eclipse plugin.</td>
</tr>
<tr>
<td>Template search locations</td>
<td>Options to append all open project directories and/or a specific template directory to your Java class path when Progress Developer Studio for OpenEdge starts.</td>
</tr>
<tr>
<td>Restore Defaults</td>
<td>Sets the value of the Customization plugin Id to com.openedge.pdt.extensibility.user and clears the Template search locations fields.</td>
</tr>
<tr>
<td>Apply</td>
<td>Saves changes and leaves the Preferences dialog open.</td>
</tr>
<tr>
<td>OK</td>
<td>Saves changes and closes the Preferences dialog.</td>
</tr>
<tr>
<td>Cancel</td>
<td>Closes the Preferences dialog without saving changes.</td>
</tr>
</tbody>
</table>

**See also**
- Template customization on page 884
- Customizing code templates on page 887
Introducing Progress DB Navigator

Progress DB Navigator is a graphical tool you can use to view the schema of a database, browse the data in tables, view and analyze application data, and issue SQL commands. DB Navigator connects to any SQL-compliant database through a JDBC driver.

If you are working with an OpenEdge database, you can use schema wizards to create, modify, and delete tables, columns, indexes, and other database objects. You can also view triggers and sequences. You cannot, however, perform administrative tasks, like dumping and loading data.

(For OpenEdge databases only) OpenEdge databases provide multi-tenant features that allow you to create, edit, and delete multi-tenant tables and sequences.

If you are working with a Microsoft SQL Server database or an Oracle database, you can view extended schema elements such as triggers, functions, and stored procedures.

The online help for DB Navigator consists of the following sections:

**Concepts**
- [DB Navigator perspective](#) on page 908
- SQL Connection profiles
- JDBC drivers
- Multitenancy (Progress OpenEdge databases only) on page 909

**Tasks**
- Common tasks for supported databases
- Working with MS SQL Server databases
- Working with OpenEdge databases
- Working with Oracle databases

**Reference**
Components of the DB Navigator perspective on page 957
MS SQL Server database
OpenEdge database
Oracle database
Preferences on page 982

For details, see the following topics:

- Concepts
- Tasks
- Reference

Concepts

DB Navigator perspective

In Eclipse, perspectives define the layout of the initial set of views in the Workbench window. DB Navigator includes the DB Navigator perspective. This perspective contains a number of different views that allow you to perform multiple activities, such as creating a SQL Connection profile used in connecting to a database, viewing database schema, and running SQL queries.

In support of these various activities, the DB Navigator perspective includes the following views:

- **Connection Info view** - A read-only display of SQL connection information for the database, shown as property/value pairs.
- **Connections view** - A display of all available SQL connection profiles and all active sessions.
- **DB Structure view** - A tree view display of the database schema structure.
- **DB Details view** - A read-only display of information specific to the selected node in the DB Structure view.
- **SQL Results view** - The results of a SQL query run from the SQL Editor.
- **SQL History view** - A list of all executed SQL queries.

The DB Navigator perspective also provides access to the SQL Editor, which allows you to write and execute SQL scripts.

(Progress Developer Studio for OpenEdge only) The DB Navigator perspective in Progress OpenEdge Developer Studio provides a toolbar button for launching OpenEdge Explorer, a tool for configuring databases. See Launching OpenEdge Explorer (Progress Developer Studio for OpenEdge only) on page 911 for more information.

See also

- SQL Connection profiles on page 909
- Accessing the DB Navigator perspective on page 911
- Launching OpenEdge Explorer (Progress Developer Studio for OpenEdge only) on page 911
- Components of the Progress DB Navigator perspective on page 957
SQL connection profiles

DB Navigator uses SQL connections defined in profiles to store information needed to connect to a database. Each profile contains all the information necessary to establish this connection, such as the specific JDBC driver to use and the connection parameters (connection name, user name, password, host name, port number, and so on).

You create a connection profile in either of the following ways:

• From the Connections view using the Add Connection Profile wizard.
• From the Window > Preferences > Progress Databases > DB Navigator Connection Profiles page.

See also
Adding SQL connection profiles on page 914
SQL Connection Profile preferences on page 984

JDBC drivers

JDBC (Java Database Connectivity) drivers allow Java-based applications to connect to SQL-compliant databases.

DB Navigator includes several JDBC drivers that allow connections to supported databases. You can see the list of all available drivers in the JDBC Drivers preferences page, which you can access by choosing Window > Preferences > Progress Databases > DB Navigator > JDBC Drivers. You can also add other JDBC drivers.

Note: Since JDBC drivers execute SQL statements, you must start databases that can service SQL clients. Also note that for an OpenEdge database, ABL access is required for trigger maintenance. Therefore, define database servers with the client type of Both 4GL and SQL (which is the default).

See also
SQL connection profiles on page 909
Configuring a JDBC driver on page 912
Connecting to a database on page 913

Multitenancy (Progress OpenEdge databases only)

Multitenancy refers to a principle in database architecture where a single instance of database runs on a server that serves multiple client organizations (tenants). DB Navigator provides multi-tenant features for OpenEdge databases.

You can use DB Navigator's multi-tenant features of to do the following:

• Create, edit, identify, and delete multi-tenant tables.
• Convert tables to multi-tenant tables.
• Create, edit, and delete multi-tenant sequences.
• Share data definition files (schema changes) across Progress OpenEdge developers.
Note: These features are limited to Progress OpenEdge databases and do not impact other databases like Oracle, SQL, etc.

See also
Working with multi-tenant tables on page 934
Working with multi-tenant sequences on page 946

Table partitioning in Progress OpenEdge databases

A partition is a division of a logical database into distinct independent parts. Database table partitioning allows you to make horizontal partitions of the table data based on a value or a range of one or more database table fields. This partitioning increases performance because the search involves limited views of data, while maintaining availability and security.

Note: A database must be enabled for table partitioning, by either using the PROUTIL command or from the OpenEdge Management console, to be able to partition the tables.

OpenEdge databases that are enabled for partitioning are categorized into two groups:
- Partitioned tables with partition policies defined on them
- Partitioned tables without any partition policies defined on them

You can use the DB Structure view to manage partitioned tables in the same way as the other OpenEdge database tables.

See also
Working with partitioned tables on page 937

Tasks

Common tasks for supported databases

This section describes the tasks that are commonly performed when you use Progress DB Navigator to work with a SQL-compliant database.

See also
Adding a SQL connection profile for an MS SQL Server database on page 920
Adding a SQL connection profile for an OpenEdge database on page 924
Adding a SQL connection profile for an Oracle database on page 952

Setting preferences

You set the preferences for Progress DB Navigator by selecting Window > Preferences. Expand the Progress DB Navigator tree view node to see the following preference options:
- **Connection Profiles** - Configure SQL connections to databases.
- **Databases>OpenEdge Schema Editing** - Set preferences associated with the schema wizards for OpenEdge databases.
- **JDBC drivers** - Configure aspects of the JDBC driver.
- **SQL Editor** - Configure SQL Editor preferences.

**See also**
- Connecting to a database on page 913
- Adding SQL connection profiles on page 914
- Configuring a JDBC driver on page 912
- SQL Editor on page 961

**Accessing the DB Navigator perspective**

To open the DB Navigator perspective:

1. Click **Open Perspective** on the **Workbench toolbar**.

   **Note:** You can also select Window > Open Perspective.

2. Select **DB Navigator** from the drop-down menu if it is listed. If it is not listed, choose **Other**. Then choose **DB Navigator** from the **Open Perspective** dialog.

   **Note:** If you have already opened the DB Navigator perspective in a workspace, you can access it again by choosing DB Navigator from the perspective bar. The location of the perspective bar is configurable. By default, the perspective bar is located on the left side of the workbench. You might need to click the chevron (>>) to expand the list of available perspectives. Alternatively, you can display more perspective tabs by dragging the tab folder to the left.

3. Connect to a database. See Connecting to a database for more information.

   **Note:** For more information on using perspectives, toolbars, and menu bars, see the Workbench User Guide in the Eclipse help system.

**Launching OpenEdge Explorer (Progress Developer Studio for OpenEdge only)**

1. To launch OpenEdge Explorer (a tool for configuring databases) from Progress OpenEdge Developer Studio's DB Navigator perspective, click the **OpenEdge Explorer** toolbar button.

2. When started, OpenEdge Explorer prompts for an administrator user name and password. The default value is admin for both user name and password. You must change the password after the initial login. After you log in, OpenEdge Explorer runs in a browser embedded in the active perspective. For online help, click one of the help buttons in OpenEdge Explorer.

3. Change the default user name and password after logging in. You can also change the default URL on the **Server preferences** page.
Configuring a JDBC driver

DB Navigator requires a JDBC driver in order to connect to a database and supplies JDBC drivers for various database types. In most cases, the driver is installed and configured automatically during installation of DB Navigator. If the driver is not configured correctly, you must edit its configuration.

Among the supplied drivers is the JDBC ODBC Bridge. Although the bridge allows you to use predefined ODBC data sources, its use is typically not recommended, as it is a less efficient connection option.

To verify that the JDBC driver you need is configured correctly and is available for a database connection, choose Window > Preferences > Progress DB Navigator > JDBC Drivers. A list of drivers appears in the JDBC Drivers preferences page.

The JDBC drivers that are already successfully configured are tagged with a blue ✓ icon. If a driver is tagged with a red ✗ icon, however, the driver is not configured. You must configure the driver before it can be loaded to connect to a database.

To configure a JDBC driver from the JDBC Drivers Preferences page:

1. Select the driver, and click Edit. The Edit Driver dialog appears.
2. Verify the driver name in the Name field and the URL in the Example URL field. This URL entry forms the default URL when you create a new SQL connection. If the URL contains an IPv6 address, be sure to include an opening bracket before and a closing bracket after the address; for example, [the-host-machine]. (See the documentation for the driver for more information.)
3. Verify that any entries listed in the Extra Class Path tab point to a valid driver. Click Add to add a driver to the list, or click Delete to delete an existing driver from the list.
4. Verify the Driver Class Name for the selected driver. You can also select the driver and then click List Drivers to retrieve a list of available names. (If no driver class name appears automatically, you can type one in the Driver Class Name field.)
5. Click OK.

Note: The listed drivers have definitions that assume that the JDBC driver classes are in the current class path or are in your JRE extensions directory. (Consult the documentation for your Java implementation for more information.) If this is not the case, you must modify the driver definition to point to the jar file or the directory that contains these classes.

6. If you successfully configure the driver, a blue ✓ icon appears next to its name in the list of drivers. If the red ✗ icon still appears next to the driver’s name, perform troubleshooting for the particular driver, as follows, to review the configuration:
   - Troubleshooting the MS SQL Server database JDBC driver
   - Troubleshooting the Progress OpenEdge database JDBC driver
   - Troubleshooting the Oracle database JDBC driver

See also
JDBC drivers on page 909
Connecting to a database

You can connect DB Navigator to multiple databases. Here is an overview of how to connect DB Navigator to a database (follow the links for more information about the tasks). Obtain or create a user account for the database. You must provide a login to access a SQL database. When you try to connect to a database in DB Navigator, you are prompted for a user ID and password. For other details about the specific database you are working with, see the relevant database documentation.

Note: (For OpenEdge databases only) If no user accounts have been established, you can use the Windows login name of the person who created the database as the user ID.

1. Configure a JDBC driver for the database. DB Navigator requires a JDBC driver in order to connect to a database and supplies JDBC drivers for various database types. In most cases, the driver is installed and configured automatically during installation of DB Navigator. If, however, the JDBC driver for your database requires driver configuration, see Configuring a JDBC driver for more troubleshooting information.

2. Start the database server. You cannot connect to a standalone database. You must start a database server before connecting to the database. To start OpenEdge databases, you can use either OpenEdge Management or OpenEdge Explorer, or the OpenEdge proserve command.

Note: Since JDBC drivers execute SQL statements, you must start databases that can service SQL clients. For OpenEdge databases, ABL access is required for trigger maintenance. Therefore, create a server with a client type of Both 4GL and SQL (which is the default).

3. Open the connection. You can select and open an existing SQL connection, or you can add a new SQL connection profile and open it. Connection profiles contain connection information, which includes, among other details, the name of the driver, a URL expression, a user name, and a password.

4. When you open a connection profile for a database, the schema for the database appears in the DB Structure view.

See also

- SQL connection profiles on page 909
- JDBC drivers on page 909
- Configuring a JDBC driver on page 912
- Adding SQL connection profiles on page 914
- Opening and closing SQL connections on page 915
- Connections view on page 958
- Connection Info view on page 958
- DB Structure view on page 959

Working with SQL connection profiles

You can add, edit, display, open, and close SQL connection profiles using the DB Navigator.

See also

- SQL connection profiles on page 909
Adding SQL connection profiles
You must have a SQL connection profile created for each MS SQL Server database, OpenEdge database, and Oracle database (or any other SQL-compliant database) whose schema you want to view in DB Navigator.

DB Navigator uses SQL connection profiles to store information needed to connect to a database. Each profile contains all the information necessary to establish this connection, such as the specific JDBC driver to use and the various connection parameters required.

Before you can add a SQL connection profile, you must have a JDBC driver for the database. Typically, the driver is installed and configured automatically during installation of DB Navigator. You may, however, find that you need to configure the driver, and you can do so from the Window > Preferences > Progress Databases > DB Navigator > JDBC Drivers preferences page.

Once you have the JDBC driver configured, you can create a SQL connection profile for your SQL-compliant database.

See also
SQL connection profiles on page 909
Configuring a JDBC driver on page 912
Connecting to a database on page 913
SQL Connection profile preferences on page 984

Editing SQL connection profiles
To edit an existing SQL connection profile in the Connections view:

1. Select the connection profile you want to edit. Right-click to display the context menu.
2. Choose Edit Connection Profile.
3. In the Edit Connection Profile dialog, modify those aspects of the connection you want to change
4. Click OK.

Note: You can click Test Connection to verify database connectivity.

See also
SQL connection profiles on page 909
Adding SQL connection profiles on page 914
SQL Connection Profile preferences on page 984

Displaying SQL connection profiles
You can access a list of SQL connection profiles by using either of the following methods:

• Select Window > Preferences > Progress Databases from the main menu bar, and then select the DB Navigator node from the preference tree. Select Connection Profiles in the expanded tree view.
• From the Connections view, expand the Connection Profiles node.

See also
SQL connection profiles on page 909
Adding SQL connection profiles on page 914
Editing SQL connection profiles on page 914
Opening and closing SQL connections on page 961
SQL Connection Profile preferences on page 984

Opening and closing SQL connections

Opening and closing SQL connections

1. To open (start) a connection:
   a) Select Window > Show View > Connections to access the Connections view in DB Navigator.
   b) Double-click a connection profile beneath the Connection Profiles node. (Or, select a connection profile and click Open Connection.) The Connection dialog appears.
   c) Enter the user ID and, optionally, a password for an account. The User and Password fields default to the values defined, if any, in the connection profile.
   d) Choose either AutoCommit or Commit On Close in the Connections view toolbar. AutoCommit, which is the default, applies changes immediately after a transaction—a SQL statement that updates the database—completes. Commit On Close saves changes and applies them when the session closes.
   e) Click OK.

   Note: When the connection is made, the connection profile name appears in the Connections view beneath the Active Connections node. The schema of the database appears in the DB Structure view.

2. To close a connection:
   a) Access the Connections view in DB Navigator.
   b) Select the connection profile of the connection you want to close, located under the Active Connections node.
   c) Click Close Connection. (Alternatively, click Close All Connections.)

See also
SQL connection profiles on page 909
SQL scripting

DB Navigator contains a SQL Editor that allows you to write and execute SQL scripts. It also has a SQL Results view that displays the results for any executed SELECT statement. You can also save SQL scripts in your project.

Starting the SQL editor

When you are connected to a database, you can use one of the following methods to start the SQL Editor:

- Select the database in the Connections view, and then click New SQL Editor from the Connections view tool bar.

  **Note:** If you create a new file in the SQL Editor, the script will not automatically be saved when you exit the editor. For new files, you will not be prompted to save the script when you exit the editor. You can save new scripts by choosing either File > Save As before you exit the editor or Save from the context menu in the SQL Editor.

- Choose Generate Select in SQL Editor from the context menu of the DB Structure view. This option is available when you have a table selected and generates a simple query of all the columns in the table. You can modify the generated code before running it.

- Choose Create Table Script from the context menu of the DB Structure view. This option is available when you have a table selected and generates a table creation script based on the schema of the selected table. You can modify the generated code before running it.

- Open an existing file from the Navigator view.

- Open an item in the SQL Editor from the New SQL File wizard. This method creates a script that is a resource in an existing project (and you are prompted to save when you exit the editor).

- Follow these steps:
  1. Choose File > New > Other.
  2. Select Progress DB Navigator > SQL Editor.
  3. Enter the name of a project folder in the Enter or select the parent folder field, and type a script filename in the File name field. The filename must be unique: You cannot enter the name of an existing file. However, you can load the contents of an existing file by selecting Link to file in the file system under the Advanced options.
  4. Click Finish.

See also

- Copying and pasting on page 917
- Generating SQL queries on page 917
- Generating a create table script on page 918
- Saving SQL scripts on page 918
Generating SQL queries

To automatically generate a SQL query for an individual table:

1. Select a table in the **DB Structure view**. Right-click to display the context menu.
2. Choose **Generate Select in SQL Editor**. The SQL Editor opens with a `SELECT` statement that contains all the columns of the table.
3. Click **Execute SQL** to execute the query. The results appear in a browser in the **SQL Results view**.

**Note:** If you select a table whose field names begin with an underscore (as is the case in a system table), you need to ensure that every field name is enclosed in quotes. Otherwise, the SQL statement fails. You can modify the **Progress DB Navigator preferences** so that **Copy To Clipboard** has the **Enclose** in single quotes option selected to ensure that all field names are quoted.

**See also**

- **Starting the SQL editor** on page 916
- **Copying and pasting** on page 917
- **Generating a create table script** on page 918
- **Saving SQL scripts** on page 918
- **SQL Editor** on page 961
- **SQL Editor Preferences** on page 986
- **SQL History view** on page 963
- **SQL Results view** on page 962

**Copying and pasting**

Copying and pasting

A **Copy Name** option is available from most context menus in the **DB Structure view** and some context menus in the **DB Details view**. **Copy Name** copies all selected node labels to the clipboard.

The **SQL Results view** contains a **Copy to Clipboard** icon in its toolbar. It also allows the data to be pasted to the clipboard.

The copied contents can be pasted into the **SQL Editor**, the **OpenEdge Editor**, or any other Windows-compliant application.

Settings in the **DB Navigator preferences** on page 982 page allow you to specify the following variations when pasting:

- **Append table name to column** - Select this option to automatically append table names to column names when they are pasted.
- **Enclose in quotes** - Select this option to automatically enclose column names in quotes when they are pasted or when generating a select in SQL Editor.
• **Delimiter** - Provide the delimiter you want to use when multiple columns are selected. (The default is a comma.)

**See also**
- Starting the SQL editor on page 916
- Generating SQL queries on page 917
- Generating a create table script on page 918
- Saving SQL scripts on page 918
- SQL Editor on page 961
- SQL Editor preferences on page 986
- SQL History view on page 963
- SQL Results view on page 962

**Generating a create table script**

When you select a table node in the DB Structure view and right-click, the context menu contains an option for running a SQL script that creates a table. When you select **Create Table Script** from the context menu, a script appears in the SQL Editor. You can use the script to generate the SQL code needed to create a copy of the schema of the selected table.

**Note:** If you try to execute the script as it is, it fails because the table already exists. At the very least, you must make the name unique for the new table before you execute the script. (For OpenEdge database tables only) Be aware that Create Table Script does not generate the extended SQL syntax for defining OpenEdge attributes such as Format or Label.

**See also**
- Starting the SQL editor on page 916
- Copying and pasting on page 917
- Generating SQL queries on page 917
- Saving SQL scripts on page 918
- SQL Editor on page 961
- SQL Editor preferences on page 986
- SQL History view on page 963
- SQL Results view on page 962

**Saving SQL scripts**

What you need to do to save scripts depends on how you start the SQL Editor.

When you start the SQL Editor from the New SQL File wizard, or when you open a SQL script that is a project resource and then you modify the script, you can save by choosing **File > Save**. The editor also prompts you to save when you exit the editor after making changes to the file.

If you start the SQL Editor in any other way, you must choose **File > Save As** before you exit the editor. You are prompted to name the file and to add it to an existing project. Then, you can save.

**See also**
- Starting the SQL editor on page 916
- Copying and pasting on page 917
- Generating a SQL query on page 917
Working with application data

To view application schema using DB Navigator, you must have READ access privileges. To edit application schema, you must have WRITE access privileges.

**Note:** (For OpenEdge databases) Users with DBA privileges can view and edit application schema.

See also

Viewing application data on page 919
Editing application data on page 919

Viewing application data

From DB Navigator, you can preview a limited set of table records and see the default arrangement of the data, or you can view all data in a table.

1. To preview a limited set of records:
   a) Select a table name in the DB Structure view.
   b) Select the Preview tab in the DB Details view.

   **Note:** By setting the Preview row limit in the DB Navigator preferences on page 982, you can adjust the number of rows displayed in the Preview tab.

2. To view all data in a table:
   a) Select a table name in the DB Structure view.
   b) Right-click on the table and choose Generate Select in SQLEditor to open a SQL editor containing the generated SELECT statement.
   c) Click Execute SQL to open a SQL Results view containing a browse with the first batch of records. To retrieve the next batch of records, click Retrieve more rows; to get all the records, click Retrieve All Rows. You can add a WHERE clause to the SQL SELECT statements to further refine the data retrieved. You can also modify the text and remove any columns.

   **Note:** By default, the system retrieves 100 records at a time. You can change this value in the Progress DB Navigator preferences.

Editing application data

To edit the data in a table:
1. Select a table name in the **DB Structure view**. Right-click to display the context menu.

2. Choose **Edit Table Data**.

   The **Editing** dialog appears, displaying the data from the selected table.

   **Note:** The browse expands automatically to the maximum size of the row.

   You can set the number of rows displayed in the **SQL Results row limits** field of the **DB Navigator preferences** on page 982. You can also filter the rows displayed by entering a **SQL WHERE** statement. For example, in this image, statement displays only those rows in which the value in the Balance column is greater than 10000. Both editing errors and SQL syntax errors appear at the bottom of the view.

   **Note:** (For OpenEdge databases only) ABL triggers do not execute when you modify data in the **Editing** dialog. For example, if you use a trigger to prompt for user confirmation before committing a change, the trigger does not execute. However, since DB Navigator uses the OpenEdge SQL interface, Java triggers do execute.

---

### Working with MS SQL Server databases

Working with an MS SQL Server database using DB Navigator allows you to set preferences, add SQL connection profiles, troubleshoot the MS SQL Server JDBC driver, and view the database schema.

#### Setting preferences

You set the preferences for Progress DB Navigator by selecting **Window > Preferences**. Expand the **Progress DB Navigator** tree view node to see the following preference options:

- **Connection Profiles** - Configure SQL connections to databases.
- **Databases>OpenEdge Schema Editing** - Set preferences associated with the schema wizards for OpenEdge databases.
- **JDBC drivers** - Configure aspects of the JDBC driver.
- **SQL Editor** - Configure SQL Editor preferences.

**See also**

- [Connecting to a database](#) on page 913
- [Adding SQL connection profiles](#) on page 914
- [Configuring a JDBC driver](#) on page 912
- [SQL Editor](#) on page 961

#### Adding a SQL connection profile for an MS SQL Server database

To add a SQL connection profile:

1. Click **Add Connection Profile** in the **Connections view**.
2. Type a unique name in the **Name** field.
3. Select the Microsoft SQL Server JDBC Driver (DataDirect) from the driver selection list. You can click New Driver if you need to configure a driver. In most cases, however, the correct driver is installed and configured automatically during installation of DB Navigator. SQL can have multiple JDBC connections. Notice that the format for the selected driver's URL appears in the URL field. If necessary, you can edit the text in the URL field to include the appropriate information for your system.

4. To edit a URL:
   a) Deselect the Specify URL Details checkbox (by default, this checkbox is selected, preventing you from editing the URL field).
   b) Supply a hostname (use localhost if the host is local to the system). If you are using an IPv6 address, insert a bracket before and after the address; for example, [thehostmachine].
   c) Type the port number.
   d) Type the database name.

   **Note:** Since the format of the URL and the required information can vary from driver to driver, see the documentation for your driver if you need more information.

5. Type the user name and, optionally, a password.

   **Note:** You must specify the user ID when creating a SQL connection.

6. Choose Open on Eclipse startup if you want the connection to open when Eclipse starts (provided the database server is already started). Note that you still must enter a valid user ID (and optionally a password). Also, be aware that this option slows the startup of Eclipse.

7. Choose Auto-Login on connect to automatically log into the database when connecting.

8. Type the name of the Host.

9. Type the correct value in the Port field.

10. Type the name of the database in the Database field.

11. Add any additional JDBC connection properties in the Additional parameters field, delimited by semicols in the format key1=value; key2=value. (Refer to the documentation for your JDBC driver for a list of supported parameters.)

12. Click Test Connection. If the connection succeeds, a Connection succeeded message appears. If the connection does not succeed, an error message appears.

13. Click OK.

**See also**

Troubleshooting the MS SQL Server JDBC driver on page 921

**Troubleshooting the MS SQL Server JDBC driver**

DB Navigator requires a JDBC driver in order to connect to a database and supplies JDBC drivers for various database types. In most cases, the driver is installed and configured automatically during installation of DB Navigator.
If the driver is configured properly, the driver is tagged with a blue ✔ icon in the JDBC Drivers preferences page. If the driver is tagged with a red ✗ icon, however, the driver is not configured and cannot be loaded to connect to a database. If the JDBC driver is not properly configured, you see an error message when you try the database connection.

To troubleshoot the JDBC driver configuration, begin by verifying that the URL specified in the database’s connection profile is correct, and then verify several other driver configuration options. These steps are as follows:

1. From the Connections view, select the database connection in the Connection Profiles list.
2. Right-click, and choose Edit Connection Profile.
3. Check the URL in the Connection Profile dialog.

**Note:** For the Microsoft SQL Server JDBC Driver (DataDirect), the URL should look similar to

```
jdbc:progress:sqlserver://hostname:port;DatabaseName=name
```

in which port is the port number of the database server and name is the logical database name. If the URL contains an IPv6 address, be sure to include an opening bracket before and a closing bracket after the address; for example, [thehostmachine].

4. Close the Edit Connection Profile dialog once you confirm that the URL is correct.
5. Choose Preferences > Window > Progress Databases > DB Navigator > JDBC Drivers to open the JDBC Drivers preferences page.
6. Select the driver you want to configure from the list, and then choose Edit. The Edit Driver dialog appears.
7. In the Edit Driver dialog:
   a) Ensure the string in the Name field is unique and indicates the correct driver.
   b) Ensure the Driver Class Name field is correct. The DataDirect Microsoft SQL Server JDBC Driver class name is as follows: `com.prgs.jdbc.sqlserver.SQLServerDriver`
   c) Click the Extra Class Path tab. Verify that the following pathnames appear:

   ```
   install_dir\oeide\architect\eclipse\plugins\com.progress.dbnavigator_2.1.0\drivers\pgsqlserver.jar
   install_dir\oeide\architect\eclipse\plugins\com.progress.dbnavigator_2.1.0\drivers\pgbase.jar
   install_dir\oeide\architect\eclipse\plugins\com.progress.dbnavigator_2.1.0\drivers\pgutil.jar
   ```

   **Note:** The `install_dir` path refers to the complete path of your product installation. You must provide this complete pathname to configure the driver properly.

d) If a required driver is not listed in the Extra Class Path list, click Add.
e) Choose the required driver; repeat for each additional driver required.
f) Use Delete to remove any incorrect pathnames.

See also
JDBC drivers on page 909
Connecting to a database on page 913
Viewing MS SQL Server database schema

You can view MS SQL server database schema details in a variety of ways.

Related Reference

MS SQL Server DB Structure view
MS SQL Server DB Details view

Displaying MS SQL Server database function details

In the DB Structure view, click any of the FUNCTION child nodes to display details about a particular function in the DB Details view. Function information includes details about the parameters; the source code; properties and their values; and a list of any dependent objects.

See also

MS SQL Server DB Structure view on page 964
MS SQL Server DB Details view on page 963

Displaying MS SQL Server database procedure details

In the DB Structure view, click any of the PROCEDURE child nodes to display details in the DB Details view about each procedure that is stored in the database and available for reuse. Procedure details include the source code as well as properties and their current values.

See also

MS SQL Server DB Structure view on page 964
MS SQL Server DB Details view on page 963

Displaying MS SQL Server database session details

In the DB Structure view, click the SESSIONS node to display session information from the sysprocesses table in the DB Details view. The sysprocesses table holds information about processes (client or system) running on MS SQL Server.

See also

MS SQL Server DB Structure view on page 964
MS SQL Server DB Details view on page 963

Displaying MS SQL Server database table details

In the DB Structure view, click any of the system table or table names to view details, such as columns, indexes, primary key, foreign key, preview, row count, and table and column privileges, in the DB Details view.

See also

MS SQL Server DB Structure view on page 964
MS SQL Server DB Details view on page 963
Displaying MS SQL Server database triggers

In the **DB Structure view**, click any of the **TRIGGER** child nodes to display trigger information in the **DB Details view**. The **DB Details view** provides details about the properties and their values as well as the trigger body source code.

**See also**

- MS SQL Server DB Structure view on page 964
- MS SQL Server DB Details view on page 963

Displaying MS SQL Server database view details

In the **DB Structure view**, the **VIEW** node lists all created views for the database. Click any of the **VIEW** child nodes to display information about a particular database view, such as columns, indexes, primary key, foreign key, preview, row count, and table or column privileges.

**See also**

- MS SQL Server DB Structure view on page 964
- MS SQL Server DB Details view on page 963

Working with OpenEdge databases

Working with an OpenEdge database using DB Navigator allows you to set preferences, add a SQL connection, troubleshoot the Progress OpenEdge JDBC driver, create a new database, work with user accounts, and view and maintain the database schema.

Setting preferences

You set the preferences for Progress DB Navigator by selecting **Window > Preferences**. Expand the **Progress DB Navigator** tree view node to see the following preference options:

- **Connection Profiles** - Configure SQL connections to databases.
- **Databases>OpenEdge Schema Editing** - Set preferences associated with the schema wizards for OpenEdge databases.
- **JDBC drivers** - Configure aspects of the JDBC driver.
- **SQL Editor** - Configure SQL Editor preferences.

**See also**

- Connecting to a database on page 913
- Adding SQL connection profiles on page 914
- Configuring a JDBC driver on page 912
- SQL Editor on page 961

Adding a SQL connection profile for an OpenEdge database

To add a SQL connection profile:
Tasks

1. Click **Add Connection Profile** in the Connections view.
2. Type a unique name in the **Name** field.
3. Select the **Progress OpenEdge JDBC Driver(DataDirect)** from the driver selection list. You can click **New Driver** if you need to configure a driver. In most cases, however, the driver is installed and configured automatically during installation of Progress DB Navigator. SQL can have multiple JDBC connections. Notice that the format for the selected driver's URL appears in the **URL** field. If necessary, you can edit the text in the **URL** field to include the appropriate information for your system.
4. To edit a URL:
   a) Deselect the **Specify URL Details** checkbox (by default, this checkbox is selected, preventing you from editing the URL field).
   b) Supply a hostname (use localhost if the host is local to the system). If you are using an IPv6 address, insert a bracket before and after the address; for example, [thehostmachine].
   c) Type the port number.
   d) Type the database name.

   **Note:** Since the format of the URL and the required information can vary from driver to driver, see the documentation for your driver if you need more information.

5. Type the user name and password. If there are authenticated users in the database, enter a valid user ID and password. These are used as the default when opening the connection. If no authenticated users exist, enter a user ID that corresponds to the Windows login name of the user who created the database.

   **Note:** You must specify the user ID when creating a SQL connection.

6. Select **Open on Eclipse startup** if you want the connection to open when Eclipse starts (provided the database server is already started).

   **Note:** You still must enter a valid user ID and password. Also, be aware that opening database connections slows the startup of Eclipse.

7. Select **Auto-Login** on connect to automatically log into the database when connecting.
8. Type the name of the Host.

   **Note:** If you are connecting to an OpenEdge Personal database (which is the default RDBMS packaged with Progress Developer Studio for OpenEdge), the host name must be **localhost**. You cannot connect to an OpenEdge Personal RDBMS database running remotely. If you are connecting to an OpenEdge Workgroup or OpenEdge Enterprise database, you can specify the host name of the system where the database server process is running. If the database is running on the local system, you can use **localhost** in place of the host name.

9. Select either **Service** or **Port**, and type the correct value. If you are unsure of the value, check the configuration settings for the database in OpenEdge Management or OpenEdge Explorer.
10. Type the logical name of the database in the **Database** field. Check for the correct name in the Databases folder in Progress Explorer.
11. Add any additional JDBC connection properties in the **Additional parameters** field, delimited by semicolons in the format `key1=value; key2=value`. 

---

Progress Developer Studio for OpenEdge : Online Help

925
12. Click **Test Connection** to verify connectivity. If the connection succeeds, a Connection succeeded message appears. If the connection does not succeed, an error message appears.

13. Click **OK**.

**Note:** See OpenEdge Data Management: SQL Development and OpenEdge Data Management: SQL Reference for more information. You can find these OpenEdge manuals in the *Product Documentation* section of the Progress Software Developer's Network Web site.

**See also**

- [SQL Connection profiles](#) on page 909
- [Working with SQL Connection profiles](#) on page 913

**Troubleshooting the Progress OpenEdge JDBC driver**

DB Navigator requires a JDBC driver in order to connect to a database and supplies JDBC drivers for various database types. In most cases, the driver is installed and configured automatically during installation of DB Navigator.

If the driver is configured properly, the driver is tagged with a blue ✓ icon in the *JDBC Drivers preferences* page. If the driver is tagged with a red ❌ icon, however, the driver is not configured and cannot be loaded to connect to a database. If the JDBC driver is not properly configured, you see an error message when you try the database connection.

To troubleshoot the JDBC driver configuration, begin by verifying that the URL specified in the database's connection profile is correct, and then verify several other driver configuration options. These steps are as follows:

1. From the **Connections view**, select the database connection in the **Connection Profiles** list.
2. Right-click, and choose **Edit Connection Profile**.
3. Check the **URL** in the **Connection Profile** dialog box.

   **Note:** For the Progress OpenEdge JDBC Driver (DataDirect), the URL should look similar to the following: `jdbc:datadirect:openedge://hostname:port;databaseName=name` in which port is the port number of the database server and name is the logical database name. If the URL contains an IPv6 address, be sure to include an opening bracket before and a closing bracket after the address; for example, `[thehostmachine]`.

4. Close the **Edit Connection Profile** dialog once you confirm that the URL is correct.
5. Choose **Windows > Preferences > Progress Databases > DB Navigator > JDBC Drivers** to open the **JDBC Drivers preferences** page.
6. Select the driver you want to configure from the list, and then choose **Edit**. The **Edit Driver** dialog appears.
7. In the **Edit Driver** dialog:
   a) Ensure the string in the **Name** field is unique and indicates the correct driver.
   b) Ensure the **Driver Class Name** field (at the bottom) is correct. The Progress OpenEdge JDBC Driver (DataDirect) class name is `com.ddtek.jdbc.openedge.OpenEdgeDriver`
c) Click the Extra Class Path tab. Verify that the following pathnames appear:

install_dir\oeide\architect\eclipse\plugins\com.progress.dbnavigator_2.1.0\drivers\openedge.jarinstall_dir\oeide\architect\eclipse\plugins\com.progress.dbnavigator_2.1.0\drivers\base.jarinstall_dir\oeide\architect\eclipse\plugins\com.progress.dbnavigator_2.1.0\drivers\util.jar

The install_dir path refers to the complete path of your product installation. You must provide this complete pathname to configure the driver properly.

d) If a required driver is not listed in the Extra Class Path list, click Add.

e) Choose the required driver; repeat for each additional driver required.

f) Use Delete to remove any incorrect pathnames.

See also

JDBC drivers on page 909
Connecting to a database on page 913

Creating an OpenEdge database

To create an OpenEdge database:

1. In the Connections view, click Menu and choose Create OpenEdge database.
2. In the Create Database dialog, enter the physical name of the database and its full path. Click Files to browse to the appropriate directory.
3. Using the radio buttons, select the type of database you want to start with.
4. Click OK. A confirmation message appears.

---

Note: To connect to the database after creating it, you must create a SQL connection profile and start a server for the database.

---

See also

SQL connection profiles on page 909
Adding SQL connection profiles on page 914
Adding a SQL connection profile for an OpenEdge database on page 924
Connections view on page 958

Working with OpenEdge database user accounts

In SQL, you must be connected to a database before issuing commands. When you use a SQL connection in DB Navigator to connect to a database, you must supply a user ID and, optionally, a password. If the user ID and password are not in the connection profile, you are prompted to provide them.

Frequently, databases do not have user accounts defined. For example, you might decide not to set up users for a local, working copy of a database that you are using for testing or development. Or, you may implement security on the application level so that database user accounts are not necessary.

If you are connecting to a database with no defined users, keep the following in mind:
You can log in with any string of characters as a user ID and, optionally, a password. However, you need a login account with database administrator (DBA) privileges if you want to create, modify, or delete database objects, or if you want to view application data.

DBA privileges are associated with the Windows login of the creator of the database. If you created the database, enter your Windows user name. No password is required.

If you are connecting to a database with defined users, contact the database administrator to set up a user account that has DBA privileges. If you are unsure which users have DBA privileges, you can check the _SYSDBAUTH system table for a list of user privileges.

**Note:** If you create a user account for an OpenEdge database that has no user accounts defined, make sure that the first account you create has DBA privileges. If the first account does not have DBA privileges, you cannot add a DBA account or any additional user accounts.

See OpenEdge Data Management: SQL Development and OpenEdge Data Management: SQL Reference for more information. You can find these OpenEdge manuals in the Product Documentation section of the Progress Software Developer's Network Web site.

### Creating OpenEdge database user accounts

To create a user account:

1. Connect to a database as a user who has DBA privileges.
3. Click **Add user to database** in the Security tab. The Create user dialog appears.
4. Type a user ID. The user ID can contain a maximum of 32 characters and cannot contain any blanks.

   **Note:** Once a user is created, you can log in to DB Navigator only with a defined user ID.

5. Type a password and confirm it. Passwords are optional. If you create a password, it can contain a maximum of 32 characters and cannot contain any blanks.

   **Note:** The password is case-sensitive; user IDs are not case-sensitive.

6. Choose to grant either DBA or RESOURCE privileges for the user.

   **Note:** If you create a user account for an OpenEdge database that has no user accounts defined, make sure that the first account you create has DBA privileges. If the first account does not have DBA privileges, you cannot add a DBA account or any additional user accounts. In SQL, DBA and RESOURCE are database-wide privileges that can be assigned to individual users. Users with DBA privileges have the ability to access, modify, or delete a database object and grant privileges to other users. RESOURCE privileges are limited compared to DBA privileges. Users with RESOURCE privileges can create database objects, but cannot modify or delete database objects. They cannot grant privileges to other users.

7. Click **Create**. This generates a SQL script that is executed and committed.

   The new user appears in the list in the Security tab and is added to the _User table.
Viewing OpenEdge database user account information

To view user account information, choose the Security node in the DB Structure view of a connected database. The Security tab appears in the DB Details view. Each row in the DB Details view represents either an authenticated user or a SQL privilege. Authenticated users are identified by the Account Type internal.

The DB Details view displays the following information for ABL-specific connections:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User ID</td>
<td>Either the user ID defined in the _User table or the user SQL authorization.</td>
</tr>
<tr>
<td>Name</td>
<td>A user name defined in an ABL tool, such as Database Administration. Since there is no way of defining a user name in SQL, the field is blank for users created in DB Navigator.</td>
</tr>
<tr>
<td>Account Type</td>
<td>The value is internal if the account is for an authenticated user found in the _User table. Otherwise, the field is blank.</td>
</tr>
<tr>
<td>DBA Rights</td>
<td>The value is Yes if the user has DBA privileges. Users with DBA privileges have the ability to access, modify, or delete a database object and to grant privileges to other users. Users with DBA privileges can use the Edit Schema wizards for maintaining the database.</td>
</tr>
<tr>
<td>Resource Rights</td>
<td>The value is Yes if the user has RESOURCE privileges. In SQL, RESOURCE privileges are limited compared to DBA privileges. Users with RESOURCE privileges can create database objects, but cannot modify or delete database objects. They cannot grant privileges to other users.</td>
</tr>
</tbody>
</table>

Note: See OpenEdge Data Management: SQL Development on the PSDN Product Documentation Web site for more information on managing SQL user accounts.

See also

Creating OpenEdge database user accounts on page 928
Changing OpenEdge database passwords on page 929
Deleting OpenEdge database user accounts on page 930
Changing OpenEdge database privileges on page 931
Granting OpenEdge database privileges on page 931

Changing OpenEdge database user account passwords

To change a password:
1. Connect to a database as a user who has DBA privileges.
2. Choose the **Security** node in the **DB Structure view**. The **Security** tab appears in the **DB Details view**.
3. Select a user account from the list in the **Security** tab. You can change passwords for any user accounts whose type is internal.
4. Click **Change user password** in the **Security** tab.
5. Type a new password, and confirm it.

**Note:** The password is case-sensitive.

6. Click **OK**.

**See also**
- Viewing OpenEdge database user account information on page 929
- Creating OpenEdge database user accounts on page 928
- Deleting OpenEdge database user accounts on page 930
- Changing OpenEdge database privileges on page 931
- Granting OpenEdge database privileges on page 931

**Deleting OpenEdge database user accounts**
You can delete any account whose user type is internal. You cannot, however, delete the account you use to connect to the database.

**To delete a user account:**
1. Connect to a database as a user who has DBA privileges.
2. Choose the **Security** node in the **DB Structure view**. The **Security** tab appears in the **DB Details view**.
3. Select one or more user accounts from the list in the **Security** tab. To select more than one row, use the **SHIFT** key to select a range of users, or use the **CTRL** key while selecting individual rows.
4. Click **Drop user** in the **Security** tab. A dialog appears in which you must confirm the deletion before it is committed.
5. Click **OK**. The user account no longer appears in the **Security** tab.

**Note:** You can view a list of users whose accounts have been deleted (and who, therefore, do not have any active privileges) by selecting the **Show Inactive Privileges** check box.

**See also**
- Viewing OpenEdge database user account information on page 929
- Creating OpenEdge database user accounts on page 928
- Changing OpenEdge database passwords on page 929
- Granting OpenEdge database privileges on page 931
- Changing OpenEdge database privileges on page 931
Granting privileges to OpenEdge database user accounts

Privileges are granted to a specific user ID. Once the user ID is established, an individual can log into the system using that user ID containing the granted privilege.

To grant privileges to a new user (one not already listed in the Security tab):

1. Connect to a database as a user who has DBA privileges.
3. Click Add SQL Privilege in the Security tab. The Add SQL Privilege dialog appears.
4. Type a user ID. The user ID can contain a maximum of 32 characters and cannot contain any blanks.
5. Choose either DBA or RESOURCE privileges for the user. In SQL, DBA and RESOURCE are database-wide privileges that can be assigned to individual users. Users with DBA privileges have the ability to access, modify, or delete a database object and to grant privileges to other users. RESOURCE privileges are limited compared to DBA privileges. Users with RESOURCE privileges can create database objects, but cannot modify or delete database objects. They cannot grant privileges to other users.
6. Click Create. This generates, executes, and commits a SQL script. The new user appears in the list in the Security tab and is an authenticated user listed in the _User table.

See also
Viewing OpenEdge database user account information on page 929
Creating OpenEdge database user accounts on page 928
Changing OpenEdge database passwords on page 929
Deleting OpenEdge database user accounts on page 930
Changing OpenEdge database privileges on page 931

Changing privileges for OpenEdge database user accounts

Changing privileges for OpenEdge database user accounts

To change privileges for a user account:

1. Connect to a database as a user who has DBA privileges.

   Note: You cannot change privileges on the account that you use to log in to the database.

3. Select a user account from the list in the Security tab.
4. Click the DBA Rights or the Resource Rights cell. A combo box appears that allows you to choose between Yes (grant privilege) and No (deny privilege).

See also
Viewing OpenEdge database user account information on page 929
Creating OpenEdge database user accounts on page 928
Changing OpenEdge database passwords on page 929
Deleting OpenEdge database user accounts on page 930
Granting OpenEdge database privileges on page 931
Viewing and maintaining OpenEdge database schema

After you connect to an OpenEdge database, DB Navigator allows you to display the OpenEdge database schema and all the schema attributes. The schema elements that you can display in the DB Structure view are tables, columns, indexes, sequences, and triggers. The schema element attributes are displayed in the DB Details view. Both views appear in the DB Navigator perspective on page 908.

DB Navigator also allows you to create, edit, delete, and identify multi-tenant tables. These multi-tenant tables are displayed in the DB Structure view along with the other tables associated with the OpenEdge database. A multi-tenant decorator added to the table icon indicates that it is a multi-tenant table. It also allows you to create, edit, and delete, multi-tenant sequences.

Working with OpenEdge database tables

In the Progress DB Navigator perspective, table names for connected databases are listed in the DB Structure view. When you select a table name (or one of its child nodes), schema details are listed in the DB Details view.

Displaying OpenEdge database table details

To display table details, click a table name (or one of the table’s child nodes) in the DB Structure view. Table information appears in the DB Details view.

In the DB Details view, the OpenEdge Table tab is displayed when you are connected to an OpenEdge database. All of the table attributes are listed alphabetically.

---

**Note:** Can-* attributes (for example Can-Read, Can-Write) are displayed only if you have DBA privileges.

See also

- Editing OpenEdge database table properties on page 933
- Adding OpenEdge database tables on page 932
- Dropping OpenEdge database tables on page 934

Adding OpenEdge database tables

To add a table to a connected database using the schema wizard:

1. Start the Add Table wizard by clicking Add table. The Add table icon is on the context menu when either the TABLE node or an individual table node is selected in the DB Structure view. It is also available as a toolbar icon on the OpenEdge Table tab of the DB Details view.
2. Type a unique name in the Table Name field.
3. (Optional) Select the Multi-tenant check box to create a multi-tenant table.
   
   **Note:** See Adding multi-tenant tables for more information.

4. In the Area field, select a storage area from the list.
   
   **Note:** To define a storage area for a multi-tenant table, you must select Support default tenant.

5. Type a unique name in the Dump File field. This name can be up to 32 characters in length.
6. Optionally, enter information in the **Label**, **Description**, **Validation**, and **Message** fields.

   **Note:** See **Add Table wizard** for more information about these fields.

7. (Optional) Select a category of tables to be displayed in the **DB structure** view by using the **Category** drop-down list.

   **Note:** See **Add Table wizard** for more information.

8. (Optional) Select the **Multi-tenant** check box to create a multi-tenant table.

   **Note:** See **Adding multi-tenant tables** for more information.

9. Select **Next** to open the **Add Columns wizard**. You must add at least one column to the table. The **Add Columns** dialog allows you to either copy columns from other tables or create a new column.

   **Note:** When you copy a column, you copy only the column schema. You do not copy data.

10. Select **Next** to **add an index** or **Finish** to complete the table and add it to the database. Since tables are displayed in ASCII order and sorting is case-sensitive, all lowercase table names sort after the uppercase table names.

    **Note:** You can also use **SQL scripting to create a table**.

**See also**

- Displaying OpenEdge database table details on page 932
- Editing OpenEdge database table properties on page 933
- Dropping OpenEdge database tables on page 934
- OpenEdge table wizard on page 968

**Editing OpenEdge database table properties**

Use the **Edit Table wizard** to modify the properties of a table.

Display the **Edit Table wizard** by clicking **Edit table**. This icon is available in the context menu (right-click) when a table name is selected in the **DB Structure view**. It is also available as a toolbar icon on the **OpenEdge Table** tab of the **DB Details view**.

**See also**

- Displaying OpenEdge database table details on page 932
- Dropping OpenEdge database tables on page 932
- Configuring a JDBC driver on page 934
- OpenEdge Edit Schema wizard on page 968
Dropping OpenEdge database tables

To delete a table:

1. Select one or more table names from the DB Structure view.
2. Right-click, and choose Drop Table from the context menu. A confirmation dialog prompts you to confirm the deletion. If you are connected to an OpenEdge database, Drop Table is also available on the OpenEdge Table tab of the DB Details view.

**Note:** To delete multiple tables simultaneously, select multiple tables in the DB Structure view, and then right-click and choose Drop Table from the context menu.

See also

- Displaying OpenEdge database table details on page 932
- Adding OpenEdge database tables on page 933
- Configuring a JDBC driver on page 932

Working with multi-tenant tables

This section describes how to work with multi-tenant tables.

**Note:** The multi-tenant features are limited to Progress OpenEdge databases and do not impact other databases like Oracle, SQL, etc.

Displaying multi-tenant table details

To display the multi-tenant table details, click a multi-tenant table name (or one of the table's child nodes) in the DB Structure view. You can also view the multi-tenant table information in the OpenEdge Table tab of the DB Details view. For a multi-tenant table, the OpenEdge Table tab of the DB Details view displays the property as Multi-tenant and the value as Yes.

The Security tab of the DB Details view displays the multi-tenant table details (like the tenant name, domain name, and the type of user). You can view details of different sets of users only if you have proper login credentials.

See also

- Multitenancy (Progress OpenEdge databases only) on page 909
- Adding multi-tenant tables on page 934
- Editing multi-tenant tables on page 935
- Identifying multi-tenant tables on page 936
- Identifying multi-tenant tables on page 936
- Deleting multi-tenant tables on page 937
- OpenEdge DB Structure view on page 967
- OpenEdge DB Details view on page 966

Adding multi-tenant tables

To add a multi-tenant table to a connected OpenEdge database using the schema wizard:
1. Click **Add table**. The Add table icon is on the context menu when either the TABLE node or an individual table node is selected in the DB Structure view. It is also available as a toolbar icon on the OpenEdge Table tab of the DB Details view.

2. Type a unique name in the **Table Name** field.

3. Select the **Multi-tenant** check box.

   **Note:** The Multi-tenant check box is unchecked and disabled when you try to add a table for database that is not enabled for multitenancy; it is unchecked and enabled when you try to add a shared table.

4. To associate the multi-tenant table with a storage area:
   a) Select the **Support default tenant** check box (if it is not already selected) to enable the **Area** field.
   b) Select a storage area from the drop-down list at the **Area** field.

   **Note:** The Support default tenant option is automatically selected when you select the Multi-tenant check box. When you clear the Support default tenant check box, the Area field gets disabled.

5. Type a unique name in the **Dump File** field. This name can be up to 32 characters in length.

6. Optionally enter information in the **Label**, **Description**, **Validation**, and **Message** fields. See Add Table wizard for more information about these fields and other options available on the wizard.

7. Select **Next** to open the Add Columns wizard. You must add at least one column to the table. The Add Columns dialog allows you to either copy columns from other tables or create a new column.

   **Note:** When you copy a column, you copy only the column schema. You do not copy data.

8. Select **Next** to add an index or **Finish** to complete the table and add it to the database.

   **Note:** Since tables are displayed in ASCII order and sorting is case-sensitive, all lowercase table names are listed after the uppercase table names.

**See also**

- Multitenancy (Progress OpenEdge databases only) on page 909
- Displaying multi-tenant table details on page 934
- Editing multi-tenant tables on page 935
- Identifying multi-tenant tables on page 936
- Converting tables to multi-tenant tables on page 936
- Deleting multi-tenant tables on page 937
- OpenEdge Table wizard on page 968
- OpenEdge DB Structure view on page 967

**Editing multi-tenant tables**

Use the Edit Table wizard to modify the properties of a multi-tenant table in the OpenEdge database.
Open the **Edit Table** wizard by selecting **Edit table**. This option is available in the context menu (right-click) when a table name is selected in the **DB Structure view**. It is also available as a toolbar icon on the **OpenEdge Table** tab of the **DB Details view**.

**Note:** When you use the **Edit Table** wizard to edit a multi-tenant table, the **Multi-tenant** check box is selected and disabled to ensure that you do not change the table to a shared table; it is unchecked and disabled when you try to edit a table for database that is not enabled for multitenancy; it is unchecked and enabled when you try to edit a shared table.

### See also
- Multitenancy (Progress OpenEdge databases only) on page 909
- Displaying multi-tenant table details on page 934
- Adding multi-tenant tables on page 934
- Identifying multi-tenant tables on page 936
- Converting tables to multi-tenant tables on page 936
- Deleting multi-tenant tables on page 937
- OpenEdge Table wizard on page 968
- OpenEdge DB Structure view on page 967

### Identifying multi-tenant tables
The **DB Structure** view in DB Navigator displays the multi-tenant tables along with the other tables associated with the OpenEdge database. A **multi-tenant decorator** which is displayed with each multi-tenant table indicates that it is a multi-tenant table.

### See also
- Multitenancy (Progress OpenEdge databases only) on page 909
- Displaying multi-tenant table details on page 934
- Adding multi-tenant tables on page 934
- Editing multi-tenant tables on page 935
- Converting tables to multi-tenant tables on page 936
- Deleting multi-tenant tables on page 937
- OpenEdge DB Structure view on page 967

### Converting tables to multi-tenant tables
1. To convert a table in the OpenEdge database to a multi-tenant table:
   a) Open the table in the **Add Table** wizard or the **Edit Table** wizard.
   b) Enter the appropriate values for the fields and select the **Multi-tenant** check box.

   **Note:** See Adding multi-tenant tables on page 934 and Editing multi-tenant tables on page 935 for more information.

2. To convert multiple tables in the database to multi-tenant tables simultaneously:
   a) Select one or more table names in the **DB Structure** view.
   b) Right-click and select **Make Table Multi-tenant** from the context menu.
Note: The Make Table Multi-tenant option is enabled when you select shared tables; it is disabled when you select multi-tenant tables or when the selected database is not enabled for multitenancy. When you select both multi-tenant tables and shared tables, the Make Table Multi-tenant option is enabled and selecting the option converts all the selected shared tables to multi-tenant tables.

See also
Multitenancy (Progress OpenEdge databases only) on page 909
Displaying multi-tenant table details on page 934
Adding multi-tenant tables on page 934
Editing multi-tenant tables on page 935
Identifying multi-tenant tables on page 936
Deleting multi-tenant tables on page 937
OpenEdge Table wizard on page 968
OpenEdge DB Structure view on page 967

Deleting multi-tenant tables
To delete multi-tenant tables in the OpenEdge database:

1. Select one or more multi-tenant table names from the DB Structure view.
2. Right-click and select Drop Table from the context menu. A confirmation dialog prompts you to confirm the deletion. The Drop Table option is also available on the OpenEdge Table tab of the DB Details view.

See also
Multitenancy (Progress OpenEdge databases only) on page 909
Displaying multi-tenant table details on page 934
Adding multi-tenant tables on page 934
Editing multi-tenant tables on page 935
Identifying multi-tenant tables on page 936
Converting tables to multi-tenant tables on page 936
OpenEdge DB Structure view on page 967
OpenEdge DB Details view on page 966

Working with partitioned tables

Identifying partitioned tables
To identify whether a table is enabled for partitioning:

1. In the DB Structure view, select the table name (or one of the table's child nodes).
2. In the DB Details view, select the OpenEdge Table tab.
   
   If the table is enabled for partitioning, the Property column displays the value of the Partitioned property as true, otherwise, as false.
You can use the other tabs of the **DB Details** view to display the details of a partitioned table. However, for partitioned tables that do not have any policies defined on them, the **Preview** and **Row Count** tabs do not display any information. To identify whether a database is enabled for table partitioning, you can use the **Database** tab of the **DB Details** node in the **DB Structure** view.

### See also
- [Table partitioning in Progress OpenEdge databases](#) on page 910
- [Working with partitioned tables](#) on page 937

### Creating partitioned tables without partition policies

You can use the **Add Table** wizard to create a partitioned table (that does not have any partition policies defined on it) with local indexes defined on it. The connected database must be enabled for table partitioning.

To add a partitioned table to a connected OpenEdge database that is enabled for table partitioning:

1. Click **Add table**. The **Add table** icon is available on the context menu when either the **TABLE** node or an individual table node is selected in the **DB Structure** view. It is also available as a toolbar icon on the **OpenEdge Table** tab of the **DB Details** view.
2. Enter a unique name in the **Table Name** field.
3. Select the **Partitioned** check box.

#### Note:
The **Partitioned** check box is enabled only for databases that are enabled for table partitioning. The **Area** field is disabled when you select the **Partitioned** check box, since partition areas are defined while defining partition rules.

4. Type a unique name in the **Dump File** field. This name can be up to 32 characters in length.
5. Optionally, enter information in the **Label**, **Description**, **Validation**, and **Message** fields. See the **OpenEdge Table wizard** for more information about these fields and other options available on the wizard.
6. Select **Next** to display the **Add Columns** page. You must add at least one column to the table. The **Add Columns** page enables you to either copy columns from other tables or create a new column.

#### Note:
When you copy a column, you copy only the column schema. You do not copy data.

7. Select **Next** to display the **Add index** page.
8. Enter a name for the index in the **Name** field. The name must be unique within the selected table.
9. Select **Global** or **Local** to specify the scope of the index.

#### Note:
The index for a partitioned table without partition policies can be either global or local. When you select **Local**, the **Area** field is disabled, since partition areas are defined while defining partition rules. A local index cannot be a word index; so, when you select **Local**, the **Word index** option is disabled.

10. Specify the other index properties (see **OpenEdge Index wizard** for detailed information) and then select **Finish** to complete the table and add it to the database.
Note: Since tables are displayed in ASCII order and sorting is case-sensitive, all lowercase table names are listed after the uppercase table names.

See also
Table partitioning in Progress OpenEdge databases on page 910
Working with partitioned tables on page 937

Working with partitioned tables with partition policies
You can use OEM or the OE SQL Editor to define partition policies on a partitioned table (that is originally created using the Add Table wizard without any partition policies defined on it). These partitioned tables are displayed along with the other OpenEdge tables in the DB Structure view.

You can use the Add index wizard to create local and global indexes for a partitioned table that has partition policy details defined on it. The index for a partitioned table can be global or local. However, the default index created for a partitioned table is a local index unless the explicitly marked as global.

By default, all the partitioned fields that are part of the partition policy details are selected for a local index. If the Global option is selected for a partitioned table, only type II areas are displayed in the Area field of the Add Index wizard. The scope of the index (Global or Local) is only applicable to partitioned tables. For tables that are not partitioned, the default index created is global. Local indexes are created only for partitioned tables and are partition-aligned indexes.

Note: A partition-aligned index is the index that has a partition key column as the leading prefix of its constraint columns.

See also
Table partitioning in Progress OpenEdge databases on page 910
Working with partitioned tables on page 937

Dropping partitioned tables
To delete a partitioned table:

1. Deallocate all the partitions defined on the table using the PROUTIL tool.
2. Select the table in the DB Structure view.
3. Right-click and select Drop Table from the context menu.
   A confirmation dialog prompts you to confirm the deletion.
   If you are connected to an OpenEdge database, Drop Table is also available on the OpenEdge Table tab of the DB Details view.

Note: To delete multiple tables simultaneously, select multiple tables in the DB Structure view, and then, right-click and select Drop Table from the context menu.

See also
Table partitioning in Progress OpenEdge databases on page 910
Working with partitioned tables on page 937
Working with OpenEdge database table columns

In the DB Navigator perspective on page 908, column names for connected databases are listed in the DB Structure view. When you select a table name (or one of its child nodes), column details are listed in the DB Details view.

Note: See OpenEdge Data Management: Database Administration in the Product Documentation section of the Progress Software Developer’s Network Web site for information about which schema modifications can (and cannot) be made online.

Displaying OpenEdge database column details

To display column details, select a table name (or one of the table's child nodes) in the DB Structure view. Column information appears in the DB Details view under the Columns tab and the OpenEdge Columns tab. The Columns tab displays all columns that conform to SQL standards. It shows SQL attributes including the SQL width, data type, size, precision, accept nulls, and comments. The OpenEdge Columns tab displays all columns and the OpenEdge attributes for each column, such as name, data type, format, label, column label, initial value, description, and others.

You can sort on any of the columns by clicking on the column label. Clicking once sorts the rows in ascending order; clicking again sorts the rows in descending order.

See also
Adding OpenEdge database columns on page 940
Editing OpenEdge database column properties on page 941
Copying OpenEdge database columns on page 941
Dropping OpenEdge database columns on page 942

Adding OpenEdge database columns

To add a column to a table:

1. Select a table name (or the Columns node under a table name) in the DB Structure view.
2. Right-click, and choose Add Column from the context menu. If you are connected to an OpenEdge database, Add Column is also available on the OpenEdge Columns tab of the DB Details view.
3. Enter column properties in the Add Columns wizard.
4. Click Add to add another column, or click Finish.

Note: Columns appear in the DB Structure view by creation date rather than by Order property value.

See also
Displaying OpenEdge database column details on page 940
Editing OpenEdge database column details on page 941
Copying OpenEdge database columns on page 941
Dropping OpenEdge database columns on page 942
OpenEdge Columns wizard on page 970
Editing OpenEdge database column details

To modify the properties of a column:

1. Select a column name from the DB Structure view.

2. Right-click, and choose Edit Column from the context menu. If you are connected to an OpenEdge database, Edit Column is also available on the OpenEdge Columns tab of the DB Details view.

3. Enter changes in the Edit Columns dialog.

   **Note:** Generally you cannot change the data type of a column, but you can change an INTEGER data type to INT64. Once you change to INT64, you cannot change back to INTEGER from DB Navigator.

4. Click Finish.

See also
Displaying OpenEdge database column details on page 940
Adding OpenEdge database columns on page 940
Copying OpenEdge database columns on page 941
Dropping OpenEdge database columns on page 942
OpenEdge Columns wizard on page 970

Copying OpenEdge database columns

To copy a column and paste it into a table:

1. Select a table name (or the Columns node under a table name) in the DB Structure view.

2. Right-click, and choose Add Column from the context menu. If you are connected to an OpenEdge database, Add Column is also available on the OpenEdge Columns tab of the DB Details view.

3. Click Copy in the Add Columns dialog.

   **Note:** You can select one or more columns and copy them to the table.

4. Select the column you want to copy in the Copy Column dialog, and then select OK. You can select one or more columns from the dialog.

   **Note:** If the column name already exists, the system displays a warning and renames the column, appending an integer to form a unique column name.

5. Click Finish. The DB Structure view is refreshed with the newly added columns.

See also
Displaying OpenEdge database column details on page 940
Editing OpenEdge database column details on page 941
Adding OpenEdge database columns on page 940
Dropping OpenEdge database columns on page 942
OpenEdge Columns wizard on page 970
Dropping OpenEdge database columns

To delete a column from a table:

1. Select one or more column names from the DB Structure view.

2. Right-click, and choose Drop Column from the context menu. A confirmation dialog prompts you to confirm the deletion. If you are connected to an OpenEdge database, Drop Column is also available on the OpenEdge Columns tab of the DB Details view.

**Note:** To delete multiple columns simultaneously, select more than one column in the DB Structure view, then right-click and select Drop Column from the context menu.

See also

- Displaying OpenEdge database column details on page 940
- Editing OpenEdge database column details on page 941
- Adding OpenEdge database columns on page 940
- Copying OpenEdge database columns on page 941

Working with OpenEdge database indexes

Indexes are used to speed up the process of searching and sorting rows. Every table can have one or more indexes.

In the DB Navigator perspective on page 908, index names for connected databases are listed in the DB Structure view. When you select a table name (or one of a table’s child nodes), index details are listed in both the Indexes and the OpenEdge Indexes tabs in the DB Details view.

Displaying OpenEdge database index details

To display index details, select a table name (or one of the table’s child nodes) in the DB Structure view.

Index information appears in the DB Details view under the Indexes tab and the OpenEdge Indexes tab. The Indexes tab displays a list of all indexes in the table that conform to SQL standards. Details in the list include the type of index; whether the index is unique; and which fields comprise the index, their ASCII order, and whether they are ascending or descending.

The OpenEdge Indexes tab displays OpenEdge-specific attributes such as primary, active, unique, word index, field type, and sort order.

See also

- Editing OpenEdge database index properties on page 943
- Adding OpenEdge database indexes on page 942
- Dropping OpenEdge database indexes on page 943
- Enabling OpenEdge database large index keys on page 944
- Index wizard on page 974

Adding OpenEdge database indexes

To add an index to a table:
1. Select a table name (or the Index node under a table name) in the DB Structure view.

2. Right-click, and choose Add Index from the context menu. If you are connected to an OpenEdge database, Add Index is also available on the OpenEdge Index tab of the DB Details view.

3. Enter index properties in the Index wizard

4. Click Finish.

See also
Displaying OpenEdge database index details on page 942
Editing OpenEdge database index properties on page 943
Dropping OpenEdge database indexes on page 943
Enabling OpenEdge database large index keys on page 944
Enabling OpenEdge database large index keys on page 944
Index wizard on page 974

Editing OpenEdge database index properties
You can make an index the primary index for the table, rename an index, or deactivate an index.

To change the component fields of an index, such as ascending or descending order or uniqueness, you must delete and then recreate the index.

To modify the properties of an index:

1. Select an index name from the DB Structure view.

2. Right-click, and choose Edit Index from the context menu. (Edit Index is also available on the OpenEdge Columns tab of the DB Details view.)

3. Enter changes in the Edit Index dialog.

4. Click Finish.

See also
Displaying OpenEdge database index details on page 942
Adding OpenEdge database indexes on page 942
Dropping OpenEdge database indexes on page 943
Enabling OpenEdge database large index keys on page 944
Index wizard on page 974

Dropping OpenEdge database indexes
While you can drop an OpenEdge database index, you cannot delete the primary index. To drop the index that is currently the primary index, you must first make another index the primary index.

To delete an index from a table:

1. Select one or more index names from the DB Structure view.

2. Right-click, and choose Drop Index from the context menu. A confirmation dialog prompts you to confirm the deletion. (Drop Index is also available on the OpenEdge Columns tab of the DB Details view.)
**Note:** To delete multiple indexes simultaneously, select multiple indexes in the **DB Structure view**, and then right-click and select **Drop Index** from the context menu.

---

**See also**
- Displaying OpenEdge database index details on page 942
- Editing OpenEdge database index properties on page 943
- Adding OpenEdge database indexes on page 942
- Enabling OpenEdge database large index keys on page 944
- Index wizard on page 974

**Enabling OpenEdge database large index keys**

Large key entries increase the amount of user data in an index from approximately 200 bytes to approximately 1970 bytes. However, even with large index keys enabled for a database, databases with 1K and 2K block sizes adhere to an entry size of approximately 200 characters.

To enable large key entries:

1. Select a table name in the **DB Structure view**.
2. Select an **Indexes** tab in the **DB Details view**.
3. Right-click, and choose **Enable Large Index Keys** from the context menu. The **DB Details view** changes to display the state of large key support. The **Enable Large Index Keys** menu option is available only if the block size is greater than 4K and if you have not previously enabled large index key support. All newly created databases have large keys enabled by default.

**Note:** Once you enable large index keys for an index, you cannot disable them.

---

**See also**
- Displaying OpenEdge database index details on page 942
- Editing OpenEdge database index properties on page 943
- Adding OpenEdge database indexes on page 942
- Dropping OpenEdge database indexes on page 943
- Index wizard on page 974

**Working with OpenEdge database sequences**

Sequences are a feature of OpenEdge databases that allow you to automatically increment a value in a field. You use sequences programmatically. Typically, when you create a new record, a procedure uses a sequence to automatically generate a sequential value for a field. For example, a sequence can increment a customer number when a new customer record is added to a table.

**Displaying OpenEdge database sequence properties**

To display sequence details, select a sequence name in the **DB Structure view**. Sequence information appears in the **DB Details view** under the **Info** tab.

The **Info** tab displays various properties and their values, including the current value.
Adding OpenEdge database sequences

To add a sequence to an OpenEdge database that is connected to Progress DB Navigator:

1. Select the SEQUENCES node from the DB Structure view.
2. Right-click, and choose Add Sequence from the context menu.
3. Enter a sequence name and properties in the Add Sequence wizard.
4. (Optional) Select the Multi-tenant check box to add a multi-tenant sequence.

   **Note:** See Adding multi-tenant sequences on page 947 for more information.

5. Click Add.
6. Click Finish.

   **Note:** The sequence is automatically added to the DB Structure view.

Editing OpenEdge database sequences

You can modify a sequence to change the limit or the sequence name. The range of sequence values includes:

- Large sequences (-9223372036854775808 to 9223372036854775807)
- Small sequences (-2147483648 to 2147483647)

To change a sequence in an OpenEdge database that is connected to Progress DB Navigator:

1. Under the SEQUENCES node in the DB Structure view, select a sequence name.
2. Right-click, and choose Edit Sequence from the context menu.
3. Change properties in the Edit Sequence wizard.
4. Click Finish.

   **Note:** See OpenEdge Data Management: Database Administration on the PSDN Product Documentation Web site for more information about sequences.
Dropping OpenEdge database sequences

To remove a sequence in an OpenEdge database that is connected to Progress DB Navigator:

1. Under the SEQUENCES node in the DB Structure view, select a sequence name.
2. Right-click, and choose Drop Sequence from the context menu. A confirmation dialog prompts you to confirm the deletion. The DB Structure view is automatically refreshed.

Note: To delete multiple sequences simultaneously, select more than one sequence in the DB Structure view, and then right-click and select Drop Sequence from the context menu.

See also
Displaying OpenEdge database sequence properties on page 944
Adding OpenEdge database sequences on page 945
Dropping OpenEdge database sequences on page 946

Working with multi-tenant sequences

This section describes how to work with multi-tenant sequences.

Note: The multi-tenant features are limited to Progress OpenEdge databases and do not impact other databases like Oracle, SQL, etc.

Displaying multi-tenant sequence properties

To display multi-tenant sequence details, select a multi-tenant sequence name in the DB Structure view. The multi-tenant sequence information appears in the DB Details view under the Info tab.

The Info tab displays various properties and their values, including the current value.

Note: The current value of the multi-tenant sequence does not display if you are connected to a database as a user with DBA privileges.

See also
Multitenancy (Progress OpenEdge databases only) on page 909
Adding multi-tenant sequences on page 947
Editing multi-tenant sequences on page 947
Deleting multi-tenant sequences on page 948
DB Structure view on page 959
DB Details view on page 960
Adding multi-tenant sequences

To add a multi-tenant sequence to an OpenEdge database that is connected to DB Navigator:

1. Select the **SEQUENCES** node from the **DB Structure view**.
2. Right-click and select **Add Sequence** from the context menu.
3. Enter a multi-tenant sequence name and properties in the **Add Sequence wizard**.
4. Select the **Multi-tenant** check box.

**Note:** The **Multi-tenant** check box is unchecked and disabled when you add a sequence for database that is not enabled for multitenancy; it is unchecked and enabled when you add a shared sequence. A shared sequence cannot be converted to a multi-tenant sequence unless it is dropped and recreated.

5. Click **Add**.
6. Click **Finish**.

**Note:** The multi-tenant sequence is automatically added to the **DB Structure view**.

See also

- **Multitenancy (Progress OpenEdge databases only)** on page 909
- **Displaying multi-tenant sequence properties** on page 946
- **Editing multi-tenant sequences** on page 947
- **Deleting multi-tenant sequences** on page 948
- **OpenEdge Add/Edit Sequence wizard** on page 976
- **DB Structure view** on page 959

Editing multi-tenant sequences

To change the properties of a multi-tenant sequence in an OpenEdge database connected to DB Navigator:

1. Under the **SEQUENCES** node in the **DB Structure view**, select a multi-tenant sequence name.
2. Right-click and select **Edit Sequence** from the context menu.
3. Change properties in the **Edit Sequence wizard**.

**Note:** When you use the **Edit Sequence** wizard to edit a multi-tenant sequence, the **Multi-tenant** check box is selected and disabled to ensure that you do not change the sequence to a shared sequence; it is unchecked and disabled when you edit a sequence for database that is not enabled for multitenancy; it is unchecked and enabled when you edit a shared sequence. Also, you cannot edit the name of a multi-tenant sequence once it is created.

4. Click **Finish**.

See also

- **Multitenancy (Progress OpenEdge databases only)** on page 909
- **Displaying multi-tenant sequence properties** on page 946
Deleting multi-tenant sequences

To delete multi-tenant sequences:

1. Select one or more multi-tenant sequence names from the DB Structure view.

2. Right-click and select Drop Sequence from the context menu. A confirmation dialog prompts you to confirm the deletion.

Note: The DB Structure view is automatically refreshed.

See also
Multitenancy (Progress OpenEdge databases only) on page 909
Displaying multi-tenant sequence properties on page 946
Adding multi-tenant sequences on page 947
Editing multi-tenant sequences on page 947
DB Structure view on page 959
DB Details view on page 960

Working with OpenEdge database triggers

Triggers are procedures that automatically execute when certain database events occur. For example, you might create a DELETE trigger that prompts the user for confirmation before a record is deleted. Triggers allow you to define custom default behaviors and business rules for your data, with the database as the central storage place for these rules.

In DB Navigator, the DB Structure view displays three types of triggers under the TRIGGERS node in the OpenEdge schema:

| 4GL TRIGGERS | The 4GLTRIGGERS node lists all the tables that contain a CREATE (C), DELETE (D), WRITE (W), FIND (F), or ASSIGN (A) trigger. In addition, an ASSIGN TRIGGERS node is located under each table node that has one or more ASSIGN triggers associated with it. |
| REPLICATION TRIGGERS | The Replication Triggers node lists all the tables that contain a REPLICATION-CREATE, REPLICATION-DELETE, or REPLICATION-WRITE trigger. Note that Replication triggers are written in ABL. Therefore, you can manage Replication Triggers in DB Navigator just like other ABL triggers. |
| Java triggers | The JAVA TRIGGERS node lists all the tables that contain INSERT, DELETE, or UPDATE Java triggers. |
Note: DB Navigator allows you to view all three types of triggers for OpenEdge databases. However, you cannot use the DB Navigator wizards to create, modify, or delete Java triggers. For more information about triggers, see the ABL manuals in the OpenEdge Product Documentation section of the Progress Communities Web site.

See also
Displaying OpenEdge database triggers on page 949
Adding OpenEdge database triggers on page 949
Editing OpenEdge database triggers on page 950
Dropping OpenEdge database triggers on page 951

Displaying OpenEdge database triggers
In the DB Structure view, click the TRIGGERS node (or any of its child nodes) to display trigger information in the DB Details view. The DB Details view shows all of the ABL triggers for the entire database.

Note: For more information, see OpenEdge Development: Programming Interfaces. You can find OpenEdge manuals in the Product Documentation section of the Progress Software Developer's Network Web site.

See also
Adding OpenEdge database triggers on page 949
Editing OpenEdge database triggers on page 950
Dropping OpenEdge database triggers on page 951

Creating OpenEdge database triggers
The Create Trigger wizard allows you to create a new trigger.

Note: You cannot create Java triggers using DB Navigator wizards.

You can create a new OpenEdge .p file, or attach an existing .p file to the trigger by performing the following steps:

1. Select the TRIGGERS, the 4GL TRIGGERS, or the REPLICATION TRIGGERS node in the DB Structure view.

2. Right-click, and choose Create Trigger from the context menu.

Note: At least one OpenEdge project must be open. If no project is open, you see an error message.

3. Enter trigger information in the Create Trigger wizard.

4. Click Finish. The new or existing file that you specified opens in the Progress Developer Studio for OpenEdge ABL Editor.

Note: For more information about triggers, see the ABL manuals in the OpenEdge Product Documentation section of the Progress Communities Web site.
See also
Displaying OpenEdge database triggers on page 949
Editing OpenEdge database triggers on page 950
Dropping OpenEdge database triggers on page 951
OpenEdge DB Structure view on page 967
OpenEdge Create/Modify Trigger wizard on page 976

Editing OpenEdge database triggers
You can edit triggers in two different ways:

• Edit the trigger code in the ABL Editor.
• Change the definition of triggers using the Edit Trigger option.

Note: You cannot modify Java triggers with the ABL Editor or the trigger wizard.

1. To edit the trigger code in the ABL Editor, open an existing trigger and edit the code (using the Progress Developer Studio for OpenEdge ABL Editor) by performing the following steps:

Note: You cannot modify Java triggers with the ABL Editor or the trigger wizard.

a) Select the trigger name in the DB Structure view.

Note: You can select either an ABL or a REPLICATION trigger. You cannot open and modify Java triggers.

b) Right-click, and choose Open in Editor from the context menu. This option opens the trigger in the ABL Editor perspective.

Note: At least one OpenEdge project must be open and the PROPATH must be set to reference the trigger location. If no project is open, you see an error message. If there is more than one project containing the trigger, the system displays a dialog and prompts you to select the trigger.

c) Edit and save the file.

2. To edit trigger properties with the Edit Trigger option, do the following:

a) Modify trigger properties by selecting a trigger name in the DB Structure view and choosing Edit Trigger from the context menu.

b) The Edit Trigger dialog opens and select whether to override triggers at run time, whether to validate triggers at run time, or attach the trigger to some other .p file.

Note: If you modify a trigger with CRC enabled, you must initiate the Set CRC option. Select the trigger, and then right-click to expose the context menu. Select Set CRC.

Note: For more information about triggers, see the ABL manuals in the OpenEdge Product Documentation section of the Progress Communities Web site.
See also
Displaying OpenEdge database triggers on page 949
Adding OpenEdge database triggers on page 949
Dropping OpenEdge database triggers on page 951
OpenEdge Create/Modify Trigger wizard on page 976
DB Structure view on page 959

Dropping OpenEdge database triggers

To delete an ABL trigger, select the trigger name in the DB Structure view and choose Drop Trigger from the context menu. A dialog prompts you to confirm the deletion.

Note: To delete multiple triggers simultaneously, select the triggers in the DB Structure view. Right-click, and then choose Drop Triggers from the context menu.

For more information, see OpenEdge Development: Programming Interfaces. You can find OpenEdge manuals in the Product Documentation section of the Progress Software Developer's Network Web site.

See also
Displaying OpenEdge database triggers on page 949
Adding SQL connection profiles on page 949
Editing OpenEdge database triggers on page 950

Importing and exporting schema changes

DB Navigator provides options to import and export data definition files (schema changes) among OpenEdge developers. You can use the Import and Export options to perform the following tasks:

- Import a data definition file.
- Export a complete data definition file.
- Import an incremental data definition file.

You can also use the Import DF and Export DF on the Database connections preferences page to import and export data definition files.

Working with Oracle databases

Working with an Oracle database using DB Navigator allows you to set preferences, add a SQL connection profile, troubleshoot the Oracle JDBC driver, and view the database schema.

Setting preferences

You set the preferences for Progress DB Navigator by selecting Window > Preferences. Expand the Progress DB Navigator tree view node to see the following preference options:

- Connection Profiles - Configure SQL connections to databases.
- Databases>OpenEdge Schema Editing - Set preferences associated with the schema wizards for OpenEdge databases.
• **JDBC drivers** - Configure aspects of the JDBC driver.
• **SQL Editor** - Configure SQL Editor preferences.

**See also**
- *Connecting to a database* on page 913
- *Adding SQL connection profiles* on page 914
- *Configuring a JDBC driver* on page 912
- *SQL Editor* on page 961

### Adding a SQL connection profile for an Oracle database

To add a SQL connection profile:

1. Select *Add Connection Profile* in the **Connections view**.
2. Type a unique name in the **Name** field.
3. Select the **Oracle JDBC Driver(DataDirect)** from the driver selection list. You can click **New Driver** if you need to configure a driver. In most cases, however, the driver is installed and configured automatically during installation of Progress DB Navigator. SQL can have multiple JDBC connections. Notice that the format for the selected driver's URL appears in the **URL** field. If necessary, you can edit the text in the **URL** field to include the appropriate information for your system.
4. To edit a URL:
   a) Deselect the **Specify URL Details** checkbox (by default, this checkbox is selected, preventing you from editing the **URL** field).
   b) Supply a hostname (use localhost if the host is local to the system).
   c) Type the port number.
   d) Type the database name.

   **Note:** Since the format of the URL and the required information can vary from driver to driver, see the documentation for your driver if you need more information.

5. Enter the user name and password.
   **Note:** You must specify the user ID when creating a SQL connection.

6. Choose **Open on Eclipse startup** if you want the connection to open when Eclipse starts (provided the database server is already started).
   **Note:** You still must enter a valid user ID and password. Also, be aware that this option slows the startup of Eclipse.

7. Choose **Auto-Login** on connect to automatically log into the database when connecting.
8. Type the name of the Host.
9. Select either Service or Port, and type the correct value.
10. Type the name of the database in the **Database** field.
11. Add any additional JDBC connection properties in the **Additional parameters** field, delimited by semicolons in the format key1=value; key2=value. (Refer to the documentation for your JDBC driver for a list of supported parameters.)

12. Click **Test Connection** to verify connectivity. If the connection succeeds, a **Connection succeeded** message appears. If the connection does not succeed, an error message appears.

13. Click **OK**.

**Troubleshooting the Oracle JDBC driver**

DB Navigator requires a JDBC driver in order to connect to a database and supplies JDBC drivers for various database types. In most cases, the driver is installed and configured automatically during installation of DB Navigator.

If the driver is configured properly, the driver is tagged with a blue ✔ icon in the **JDBC Drivers preferences** page. If the driver is tagged with a red 🔄 icon, however, the driver is not configured and cannot be loaded to connect to a database. If the JDBC driver is not properly configured, you see an error message when you try the database connection.

To troubleshoot the JDBC driver configuration, begin by verifying that the URL specified in the database's connection profile is correct, and then verify several other driver configuration options. These steps are as follows:

1. From the **Connections view**, select the database connection in the **Connection Profiles** list.

2. Right-click, and choose **Edit Connection Profile**.

3. Check the **URL** in the **Connection Profile** dialog box.

   **Note:** For the Oracle JDBC Driver (DataDirect), the URL should look similar to `jdbc:progress:oracle://hostname:port:databaseName=name`, in which port is the port number of the database server and name is the logical database name.

4. Close the **Edit Connection Profile** dialog once you confirm that the URL is correct.

5. Choose **Windows > Preferences > Progress Databases > DB Navigator > JDBC Drivers** to open the **JDBC Drivers preferences** page.

6. Select the driver you want to configure from the list, and then choose **Edit**. The **Edit Driver** dialog appears.

7. In the **Edit Driver** dialog:
   a) Ensure the string in the **Name** field is unique and indicates the correct driver.
   b) Ensure the **Driver Class Name** field is correct. The Oracle JDBC Driver (DataDirect) class name is `com.prgs.jdbc.oracle.OracleDriver`
   c) Click the **Extra Class Path** tab. Verify that the following pathnames appear:

      ```
      install_dir\oeide\architect\eclipse\plugins\com.progress.dbnavigator_2.1.0\drivers\pgoracle.jar
      install_dir\oeide\architect\eclipse\plugins\com.progress.dbnavigator_2.1.0\drivers\pgbase.jar
      install_dir\oeide\architect\eclipse\plugins\com.progress.dbnavigator_2.1.0\drivers\pgutil.jar
      ```

      The **install_dir** path refers to the complete path of your product installation. You must provide this complete pathname to configure the driver properly.

8. If a required driver is not listed in the **Extra Class Path** list, click **Add**.
9. Choose the required driver; repeat for each additional driver required.
10. Use Delete to remove any incorrect pathnames.

**See also**
JDBC drivers on page 909
Connecting to a database on page 913

**Viewing Oracle database schema**
You can display the Oracle database schema in a variety of ways.

**See also**
Oracle DB Structure view on page 980
Oracle DB Details view on page 978

**Displaying Oracle database function details**
In the DB Structure view, click any of the FUNCTION child nodes to display details about a particular function in the DB Details view. Function information includes details about the parameters; the source code; properties and their values; and a list of any dependent objects.

**See also**
Oracle DB Structure view on page 980
Oracle DB Details view on page 978

**Displaying Oracle database jobs details**
In the DB Structure view, click the Jobs child node of the Monitor parent node to display information about a job, or task, that you run once or on a particular schedule, in the DB Details view.

**See also**
Oracle DB Structure view on page 980
Oracle DB Details view on page 978

**Displaying Oracle database package body details**
In the DB Structure view, click any of the PACKAGE BODY child nodes to display details about a particular package body in the DB Details view. Details include the source code as well as properties and their values.

**See also**
Oracle DB Structure view on page 980
Oracle DB Details view on page 978

**Displaying Oracle database package details**
In the DB Structure view, click any of the PACKAGE child nodes to display details about a particular package in the DB Details view. Details include the source code; properties and their values; and a list of any dependent objects.
See also
Oracle DB Structure view on page 980
Oracle DB Details view on page 978

Displaying Oracle database procedure details
In the DB Structure view, click any of the PROCEDURE child nodes to display details in the DB Details view about each procedure that is stored in the database and available for reuse. Procedure details include parameters; the source code; properties and their current values; and a list of any dependent objects.

See also
Oracle DB Structure view on page 980
Oracle DB Details view on page 978

Displaying Oracle database sequence properties
In the DB Structure view, click any of the SEQUENCE child nodes to display details about a particular sequence in the DB Details view. Sequence information appears in the DB Details view under the Info and Details tabs, which display various properties and their current values.

See also
Oracle DB Structure view on page 980
Oracle DB Details view on page 978

Displaying Oracle database sessions details
In the DB Structure view, you can view two different types of session details.

From the Database node, you can click the SESSIONS node to display information about the current user-to-database connection in the DB Details view. From the Monitor node, you can click the Sessions node to see information about the current session, such as username, Session ID, machine name, and the user’s logon time, in the DB Details view.

See also
Oracle DB Structure view on page 980
Oracle DB Details view on page 978

Displaying Oracle database system parameter details
In the DB Structure view, click the System Parameters child node of the Instance parent node to display all system parameters (retrieved from the v$system_parameter table) in the DB Details view.

See also
Oracle DB Structure view on page 980
Oracle DB Details view on page 978
Displaying Oracle database table details

In the DB Structure view, click any of the table names to view details, such as columns, indexes, primary key, foreign key, preview, row count, table and column privileges, a list of dependent objects, and property status, in the DB Details view.

See also
Oracle DB Structure view on page 980
Oracle DB Details view on page 978

Displaying Oracle database TableSpace details

In the DB Structure view, click the TableSpace child node of the Instance parent node to display all table spaces, or logical storage units, in the database.

See also
Oracle DB Structure view on page 980
Oracle DB Details view on page 978

Displaying Oracle database Top SQL details

In the DB Structure view, click the Top SQL child node of the Monitor parent node to display SQL executions in the DB Details view, enabling you to see how resources are affected when a SQL statement is executing.

See also
Oracle DB Structure view on page 980
Oracle DB Details view on page 978

Displaying Oracle database triggers

In the DB Structure view, click any of the TRIGGER child nodes to display trigger information in the DB Details view. The DB Details view provides details about the properties and their values as well as the trigger body source code.

See also
Oracle DB Structure view on page 980
Oracle DB Details view on page 978

Displaying Oracle database Users details

In the DB Structure view, click the Users child node of the Security parent node to display all authenticated users by username, user ID, and the date on which the user was created, in the DB Details view.

See also
Oracle DB Structure view on page 980
Oracle DB Details view on page 978
Displaying Oracle database view details

In the DB Structure view, the VIEW node lists all created views for the database. Click any of the VIEW child nodes to display information about a particular database view, such as columns, indexes, primary key, foreign key, preview, row count, privileges, a list of dependent objects, and status, in the DB Details view.

See also
Oracle DB Structure view on page 980
Oracle DB Details view on page 978

Reference

Components of the DB Navigator perspective

The topics in this section describe the views, wizards, and dialogs associated with the default DB Navigator perspective.

Connection dialog

The Connection dialog box appears when you:

- Select a connection profile in the Connections view.
- Choose Open connection from the context (right-click) menu, double-click a connection profile, or click Open connection.

Clicking OK in the Connection dialog creates a connection between DB Navigator and a database.

Note: The Open Connection dialog appears only when the Auto-Login on connect option is not selected.

The Connection dialog contains a read-only display of the values defined for the profile name, the driver, and the URL. It also contains the user ID and password, if they were specified in the connection profile. However, you can override the user ID and password in the User and Password fields.

In addition, the Connection dialog allows you to select one of these settings:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutoCommit</td>
<td>Any changes to the database are committed immediately. This mode is selected by default.</td>
</tr>
<tr>
<td>Commit On Close</td>
<td>Any changes to the database are committed when you either close the session or select Commit.</td>
</tr>
</tbody>
</table>
Note: If neither AutoCommit nor Commit On Close is selected, you must click Commit in the Connections view toolbar to commit any changes that you have executed from the SQL Editor. If you perform schema management operations using the SQL Editor with AutoCommit turned off, there will be pending SQL transactions. If the connection is closed or severed, the transaction is automatically rolled back. Therefore, you should always select Commit On Close if you deselect AutoCommit. The OpenEdge schema wizard automatically commits SQL transactions even if AutoCommit is not selected.

See also
Connections view on page 958

Connection Info view
The Connection Info view displays SQL connection information for the database selected in the Connections view. Connection information is shown as property/value pairs.

Connections view
The Connections view contains a tree view that displays all available SQL connection profiles and all active sessions.

Note: Several of the controls described in the following table are also available from a context menu. To access this menu, select a connection profile in the Connections view, and right-click.

The Connections view contains the following toolbar buttons:

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Icon" /></td>
<td><strong>Add Connection Profile</strong> displays the Add Connection Profile dialog.</td>
</tr>
<tr>
<td><img src="image2.png" alt="Icon" /></td>
<td><strong>Open Connection</strong> opens the Connection dialog for the connection profile selected in the tree view.</td>
</tr>
<tr>
<td><img src="image3.png" alt="Icon" /></td>
<td><strong>New SQL Editor</strong> opens the SQL Editor.</td>
</tr>
<tr>
<td><img src="image4.png" alt="Icon" /></td>
<td><strong>Close All Connections</strong> closes all open connections and disables any views that have related information.</td>
</tr>
<tr>
<td><img src="image5.png" alt="Icon" /></td>
<td><strong>Close Connection</strong> closes the connection that is selected in the Connections view.</td>
</tr>
<tr>
<td><img src="image6.png" alt="Icon" /></td>
<td><strong>Rollback</strong> rolls back a transaction. Any changes made by the transaction are undone and the database is restored to its initial state. Rollback is disabled if the database connection was started with AutoCommit turned on.</td>
</tr>
<tr>
<td><img src="image7.png" alt="Icon" /></td>
<td><strong>Commit</strong> makes changes made by a transaction permanent. Commit is disabled if the database connection was started with AutoCommit turned on.</td>
</tr>
<tr>
<td><img src="image8.png" alt="Icon" /></td>
<td>Collapses the display of all hierarchical elements in the list, leaving only the top-level entities visible.</td>
</tr>
<tr>
<td><img src="image9.png" alt="Icon" /></td>
<td><strong>Create OpenEdge database</strong> opens the Create database dialog.</td>
</tr>
</tbody>
</table>
See also
Connecting to a database on page 913
Connection dialog on page 957
Connection Info view on page 958

DB Structure view

When you open a connection in the Connections view, the schema for the database appears in the DB Structure view. The DB Structure appears as a tree view and begins with the root node Database. Depending on the particular SQL-compliant database you are working with, the nodes that appear in the tree view might be different.

Note: You can configure DB Navigator to display or hide schemas. By default, catalogs are displayed while schemas are hidden from view. See DB Navigator preferences on page 982 for more information.

When you select certain nodes in the DB Structure view, properties and values appear in the DB Details view.

When you right-click a node in the DB Structure view, a context menu appears that allows you to view schema; for OpenEdge databases, you can view, create, and edit schema. The context menu that appears is dependent on the type of database whose structure you are viewing and the node that is currently selected.

(For OpenEdge databases only) To convert tables in the OpenEdge database to multi-tenant tables, use the context menu Make Table Multi-tenant option. See Converting tables to multi-tenant tables on page 936.

The DB Structure view contains a toolbar button that launches the DB Structure Filters dialog. The DB Structure Filters dialog allows you to hide schema, catalogs, and table types. You can specify a delimited list of patterns to exclude or include. For example, you can enter C*, D* to display only those tables starting with c or d. To display those tables starting with the letters between A and F, enter [A-F]*.

You can also hide tables and columns based on a regular expression pattern. A regular expression is a mechanism for describing text patterns when executing the filter. These expressions are made up of ordinary characters.

The DB Structure Filters dialog also allows you to select the category of tables that you want to view in the DB Structure view. You can either choose from a list of existing categories or create a new one.

Note: Your choices in the DB Structure Filters dialog are saved with the connection profile.

The root node Database is at the top level of the DB Structure view. If the database supports catalogs and you have selected Show catalog in the DB Navigator preferences, all catalogs appear below the database node. If you have selected Show schema in the DB Navigator preferences, all schemas appear below the database node. Beneath the schema and catalog nodes, the following child nodes appear:

| SYNONYM | These SQL elements display all synonyms that were created in SQL against tables. A synonym is an alias that SQL statements can use instead of the name specified when the table, view, or synonym was created. The child nodes contain only tables, views, or other synonyms. They do not display columns or indexes of the tables. |
| SYSTEM TABLE | Displays virtual systems tables and other meta-schema tables. These tables cannot be altered. |
### TABLE
Displays all application tables. Table nodes expand to show columns and indexes. When you select a table name, data and meta-data appear in the **DB Details view**.

**Note:** Since tables are displayed in ASCII order and sorting is case-sensitive, all lowercase table names are listed after the uppercase table names.

### VIEW
Displays the names of all created views. A SQL view allows you to extract rows and columns from one or more related tables and to treat the resulting data set as a table that can be viewed or updated.

Depending on which SQL-compliant database's structure you are viewing, the remaining nodes shown might vary.

**See also**
- SQL scripting on page 916
- DB Navigator preferences on page 982
- MS SQL Server DB Structure view on page 964
- OpenEdge DB Structure view on page 967
- Oracle DB Structure view on page 980
- DB Details view on page 960
- Connections view on page 958

### DB Details view
The **DB Details** view is a read-only display of information specific to the selected node in the **DB Structure view**. You can access this detailed view through the context menu for the selected node.

The context menu that appears is dependent on the type of database whose structure you are viewing and the node that is currently selected in the **DB Structure view**. When a table node (or a child node) is selected, the following tab folders appear in the DB Details view:

<table>
<thead>
<tr>
<th>Columns</th>
<th>Displays a list of all columns with related SQL information.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indexes</td>
<td>Displays a list of indexes in the selected table.</td>
</tr>
<tr>
<td>Primary Key</td>
<td>Displays the primary keys defined for the selected table.</td>
</tr>
<tr>
<td></td>
<td>This is relevant only to tables created through SQL, although it displays the components of the OpenEdge primary key.</td>
</tr>
<tr>
<td>Foreign Key</td>
<td>Displays any foreign key restraints defined for a selected SQL table.</td>
</tr>
<tr>
<td>Preview</td>
<td>Displays a subset of the records in the selected table.</td>
</tr>
<tr>
<td></td>
<td>In the <strong>Preferences</strong>, you can set the number of rows returned.</td>
</tr>
</tbody>
</table>
Row Count | Displays the total number of rows in the selected table.

**Note:** If you have an OpenEdge database table that contains many records (over a million, for example), it is not advisable to select this panel. Since OpenEdge manually counts the records, which can take a long time, the system appears to freeze.

| Privileges | Displays both table and column privileges (as a toggle). Includes the columns Table, Column, Privilege, Grantor, Grantee, and Grantable.

Depending on the type of database whose details you are viewing, there may be other tab folders (in addition to the ones described here) in the **DB Details** view.

**See also**

- [DB Structure view](#) on page 959
- [MS SQL Server DB Details view](#) on page 963
- [OpenEdge DB Details view](#) on page 966
- [Oracle DB Details view](#) on page 978

**SQL Editor**

The **SQL Editor** allows you to view, write, and execute SQL scripts.

In the edit area, you can enter one or multiple SQL statements. Multiple statements are delimited with a semicolon. You can modify the delimiter in the [Progress DB Navigator preferences](#).

If you enter a dot after a schema name or a catalog name, the auto-completion menu appears showing all tables. A secondary menu shows all columns within that table. A menu of all columns is displayed upon entering a dot after the table.

The toolbar of the editor contains the following controls:

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="#" alt="Execute" /></td>
<td>Execute the SQL statement(s). <strong>Note:</strong> The Execute icon is disabled until you select a database.</td>
</tr>
<tr>
<td><img src="#" alt="Select File" /></td>
<td>Select and open an existing .sql file.</td>
</tr>
<tr>
<td><img src="#" alt="Clear" /></td>
<td>Clear all content from the editor.</td>
</tr>
<tr>
<td><img src="#" alt="Connection" /></td>
<td>Select a database connection. When you are creating a new script, the default is the connection that is selected in the <strong>Connections</strong> view. <strong>Note:</strong> If more than one catalog exists, another drop-down list appears in which you specify which catalog to use.</td>
</tr>
</tbody>
</table>
You can choose to enable auto-completion assistance for columns within the SQL Editor. Enabling auto-completion allows you to type any letter in the SQL Editor, and then type the CTRL+SPACE keyboard combination to see if there are multiple words that match. If the letter you type has only one match, that key word is entered into the editor.

To enable auto-completion for columns, select the **Columns auto-completing assistance** option from the **Progress DB Navigator preferences** page.

**Note:** Be aware that auto-completion can slow performance during connection.

**See also**

- Starting the SQL editor on page 916
- Copying and pasting on page 917
- Generating a SQL query on page 917
- Generating a create table script on page 918
- Saving SQL scripts on page 918
- SQL Editor Preferences on page 986
- Setting database preferences on page 910
- SQL History view on page 963
- SQL Results view on page 962

**SQL Results view**

When you run a SQL query from the SQL editor, a result set appears in the **SQL Results** view. Result sets for each subsequent execution appear in the same view, but in a separate folder. **Folder** tabs are numbered consecutively. If any errors occur, the SQL error is displayed in a message dialog that includes the error code.

The toolbar contains the following controls:

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="icon" alt="Retrieve next result set" /></td>
<td>Retrieve the next result set. By default, the result set is batched in groups of 100 records. The next batch of records can be added to the view by clicking this control. <strong>Note:</strong> You can set the SQL Results rows to batch value in the <strong>DB Navigator preferences</strong> on page 982 page.</td>
</tr>
<tr>
<td><img src="icon" alt="Retrieve all rows" /></td>
<td>Retrieve all rows. Expands the number of rows in a result set to include the complete output of a query.</td>
</tr>
<tr>
<td><img src="icon" alt="Copy all rows" /></td>
<td>Copy all rows in the view to the clipboard.</td>
</tr>
<tr>
<td><img src="icon" alt="Close tab folder" /></td>
<td>Close the tab folder.</td>
</tr>
</tbody>
</table>

The status bar at the bottom of the folder shows the number of the row that is selected. It also indicates the number of rows in the result set.

**See also**

- Starting the SQL editor on page 916
- SQL Editor on page 961
- SQL History view on page 963
SQL History view

The SQL History view displays a list of all executed unique SQL queries for the current session. It does not persist the queries. The context menu contains three options:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open in editor</td>
<td>Opens the selected query in a new editor.</td>
</tr>
<tr>
<td>Remove from history</td>
<td>Deletes the selected query from the list.</td>
</tr>
<tr>
<td>Copy to Clipboard</td>
<td>Copies the selected query to the Clipboard.</td>
</tr>
</tbody>
</table>

See also

Starting the SQL editor on page 916
SQL Editor on page 961
SQL Results view on page 962

MS SQL Server database

You can view MS SQL Server database schema by using a Details view and a Structure view.

MS SQL Server DB Details view

The DB Details view is a read-only display of information specific to the selected node in the DB Structure view. You can access this detailed view through the context menu for the selected node.

Note: This detailed view is available only for certain nodes. If the detailed view is unavailable for a node you select in the DB Structure view, no information appears in the DB Details view.

The content of the DB Details view changes depending on which node you select in the DB Structure view. When you select a child of one of the nodes listed in the left column of the following table, the corresponding information appears in the DB Details view:

<table>
<thead>
<tr>
<th>SYSTEM TABLE or TABLE or VIEW</th>
<th>For a child node, displays details about the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Columns - A list of all columns and related SQL information. You can sort the column names in ascending or descending order by clicking the Column label.</td>
</tr>
<tr>
<td></td>
<td>• Indexes - A list of indexes in the selected table.</td>
</tr>
<tr>
<td></td>
<td>• Primary key - A list of the primary keys defined for the selected table.</td>
</tr>
<tr>
<td></td>
<td>• Foreign key - A list of any foreign key restraints defined for a selected SQL table.</td>
</tr>
<tr>
<td></td>
<td>• Preview - A subset of the records in the selected table. You can set the number of rows returned in the Preferences page.</td>
</tr>
<tr>
<td></td>
<td>• Row count - The total number of rows in the selected table.</td>
</tr>
<tr>
<td></td>
<td>• Privileges - A list of both table and column privileges (as a toggle).</td>
</tr>
<tr>
<td>PROCEDURE or TRIGGER</td>
<td>For a child node, displays the SQL code in the Source tab and property/value details, such as date created, in the Info tab.</td>
</tr>
</tbody>
</table>
FUNCTIONS | Displays the SQL code in the Source tab and property/value details, such as date created, in the Info tab.

SESSIONS | Displays details about the current schema.

**MS SQL Server DB Structure View**

When you open a connection in the Connections view, the schema for the database appears in the DB Structure view. The DB Structure view is a tree view that shows a separate structure for each schema under the root node Database.

**Note:** You can configure DB Navigator to display or hide schemas. By default, catalogs are displayed while schemas are hidden from view. See DB Navigator preferences on page 982 for more information.

When you select certain nodes in the DB Structure view, properties and values appear in the DB Details view. The top-level nodes in the MS SQL Server DB Structure view are Database and Monitor. You can use filters to show only certain schema items in the view.

**Database**

The root node Database is at the top level of the DB Structure view. If the database supports catalogs and you have selected Show catalog in the DB Navigator preferences, all catalogs appear below the database node. If you have selected Show schema in the DB Navigator preferences, all schemas appear below the database node.

Beneath the schema and catalog nodes, the following child nodes appear:

<table>
<thead>
<tr>
<th>SYSTEM TABLE</th>
<th>Displays virtual systems tables and other metaschema tables. These tables cannot be altered.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE</td>
<td>Displays all application tables. Table nodes expand to show columns, indexes, and triggers. When you select a table name, data and meta-data appear in the DB Details view.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Since tables are displayed in ASCII order and sorting is case-sensitive, all lowercase table names sort after the uppercase table names.</td>
</tr>
<tr>
<td>VIEW</td>
<td>Displays the names of all created views.</td>
</tr>
<tr>
<td>PROCEDURE</td>
<td>Displays all procedures that are stored in the database and available for reuse.</td>
</tr>
<tr>
<td>TRIGGER</td>
<td>Displays all triggers, which execute in response to a predefined event or activity.</td>
</tr>
<tr>
<td>FUNCTIONS</td>
<td>Displays all functions, or pieces of a program that you can use separately from the remainder of the program.</td>
</tr>
<tr>
<td>SESSIONS</td>
<td>Displays the session information for the current catalog.</td>
</tr>
</tbody>
</table>

**Monitor**
Below the database node is a Monitor node that contains the following subnode:

| SECTIONS | Displays the information for the current database session. |

**See also**
MS SQL Server DB Details view on page 963

**SQL terminology for ABL programmers**

This section is for the ABL (Advanced Business Language) programmer who is not familiar with SQL. It describes how some SQL terms and concepts differ from ABL.

**Definition**

SQL is the language used to communicate with and extract information from a SQL-compliant database. Using SQL you can read, write, and remove information from a database using English-like statements.

**Rows and columns**

A SQL table is a group of related data composed of rows and columns. The term row is equivalent to the ABL term record. The term column is equivalent to the ABL term field.

**SQL column widths**

An OpenEdge database can contain columns of variable length. However, SQL CREATE TABLE statements specify the maximum width of each column in a table. Although ABL programs have the ability to insert data whose length exceeds the maximum width, SQL applications are not able to read a row if a column contains data greater than the maximum width.

**Note:** OpenEdge contains a command-line tool called DBTOOLS that allows you to fix column widths. See OpenEdge Data Management: Database Administration in the Product Documentation section of the Progress Software Developer's Network Web site.

**Schema**

In OpenEdge databases, a schema is defined as the area in which all system and user information is stored. An OpenEdge database viewed from SQL contains a schema area, referred to as the PUB (Public) schema. It also contains a schema called SYSPROGRESS, which stores system catalog tables. In SQL, a schema is a collection of related database objects, such as tables or views. A SQL database can contain several schemas.

**Note:** See OpenEdge Data Management: SQL Development in the Product Documentation section of the Progress Software Developer's Network Web site for information about the compatibility of ABL and SQL.

**OpenEdge database**

You can view OpenEdge database schema using a Details view and a Structure view.
**OpenEdge DB Details view**

The **DB Details** view is a read-only display of information specific to the selected node in the **DB Structure view**.

**Note**: This detailed view is available only for certain nodes. If the detailed view is not available for a node you select in the **DB Structure view**, no information appears in the **DB Details view**.

The content of the **DB Details** view changes depending on which node you select in the **DB Structure view**. There are two top-level nodes: Database and Security.

**Database**

When you select a child of one of the nodes listed in the left column of the following table, the corresponding information appears in the **DB Details view**:

<table>
<thead>
<tr>
<th>SYSTEM TABLE or TABLE</th>
<th>For a child node, displays details about the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Columns - A list of all columns and related SQL information.</td>
</tr>
<tr>
<td></td>
<td>• Indexes - A list of indexes in the selected table.</td>
</tr>
<tr>
<td></td>
<td>• Primary key - A list of the primary keys defined for the selected table. This is relevant only to tables created through SQL, although it will display the components of the OpenEdge primary key.</td>
</tr>
<tr>
<td></td>
<td>• Foreign key - A list of any foreign key restraints defined for a selected SQL table.</td>
</tr>
<tr>
<td></td>
<td>• Preview - A subset of the records in the selected table. You can set the number of rows returned in the Preferences page.</td>
</tr>
<tr>
<td></td>
<td>• Row count - The total number of rows in the selected table.</td>
</tr>
</tbody>
</table>

**Note**: If you have an OpenEdge database table that contains many records (over a million, for example), it is not advisable to select this panel. Since OpenEdge manually counts the records, which can take a long time, the system appears to freeze.

<table>
<thead>
<tr>
<th>SYSTEM TABLE or TABLE</th>
<th>For a child node, displays details about the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Privileges - A list of both table and column privileges (as a toggle).</td>
</tr>
<tr>
<td></td>
<td>• Partition Key - A list of partition key attributes in the selected database table.</td>
</tr>
<tr>
<td></td>
<td>• OpenEdge Table - A list of table attributes. Attributes are extracted from the _File system table.</td>
</tr>
</tbody>
</table>

**Note**: The Area and Can-* fields are available only for users who have DBA privileges.

<table>
<thead>
<tr>
<th>VIEW</th>
<th>For a child node, displays view properties, which are column name, data type, size, decimal, default value, accept null value, and comments.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• OpenEdge Columns - A list of column attributes. Attributes are extracted from the _Field system table. The columns can be sorted in ascending or descending order by clicking the Name label.</td>
</tr>
<tr>
<td></td>
<td>• OpenEdge Indexes - A list of index attributes.</td>
</tr>
</tbody>
</table>
For each child node, displays the properties of the sequence, including its current value.

For each child node, displays details for a trigger, including Trigger Type, Table, Column, Event, Procedure, Overridable, and Check CRC.

Displays database properties under the Database tab. Under the Tables & Columns tab, it displays all columns from all tables and their properties sorted by table and column names. You can sort the table names in ascending or descending order by clicking the Column label.

### Security

When you select the **Security** node, the **DB Details** view displays the **Security** tab. The **Security** tab displays information from the _Users and the _Sysdbauth tables regarding user accounts and privileges. You can manage user accounts from the icons on the **Security** tab in the **DB Details** view. For multi-tenant tables in the OpenEdge database, the **Security** tab displays information from the _Tenant table. This information includes the tenant name, domain name, and the type of user.

### OpenEdge DB Structure view

When you open a connection in the **Connections view**, the schema for the database appears in the **DB Structure** view. The **DB Structure** view is a tree view that shows a separate structure for each schema under the root node Database. Depending on the database, the root and other nodes that appear in the tree view might be different.

**Note:** You can configure DB Navigator to display or hide schemas. By default, catalogs are displayed while schemas are hidden from view. See [DB Navigator preferences](#) on page 982 for more information.

When you select certain nodes in the **DB Structure** view, properties and values appear in the **DB Details** view. The top-level nodes in the **OpenEdge DB Structure** view are Database and Security. You can use context menus to work with the schema, and you can use filters to show only certain schema items in the view.

### Database

The root node **Database** is at the top level of the **DB Structure** view. If the database supports catalogs and you have selected **Show catalog** in the DB Navigator preferences, all catalogs appear below the database node. If you have selected **Show schema** in the DB Navigator preferences, all schemas appear below the database node.

Beneath the schema and catalog nodes, the following child nodes appear:

| SYNONYM | These SQL elements display all synonyms that were created in SQL against tables. The child nodes contain only tables, views, or other synonyms. They do not display columns or indexes of the tables. |
| SYSTEM TABLE | Displays virtual systems tables and other metaschema tables. These tables cannot be altered. |
**TABLE**
Displays all application tables. Table nodes expand to show columns, indexes, and triggers.

---
**Note:** Since tables are displayed in ASCII order and sorting is case-sensitive, all lowercase table names are listed after the uppercase table names.

This node also displays the multi-tenant tables associated with the OpenEdge database. A multi-tenant decorator (†) which is displayed with each multi-tenant table indicates that it is multi-tenant table.

**VIEW**
Displays the names of all created views. A SQL view allows you to extract rows and columns from one or more related tables and to treat the resulting data set as a table that can be viewed or updated.

**SEQUENCES**
Displays sequences in alphabetical order. A sequence is an incremented number series applied to a particular field (column). Whenever a new record (row) is created, the number is incremented based on the value in the previous record.

**TRIGGERS**
Displays ABL schema triggers, replication triggers, and Java triggers.

**DB DETAILS**
Displays database, table, and column properties in the DB Details view.

---

**Security**
When you select the **Security** node in the **DB Structure** view, a list of user accounts appears in the **DB Details** view. You can manage user accounts from the icons on the **Security** tab in the **DB Details** view.

**OpenEdge edit schema wizards**
The **Edit Schema** wizards allow you to modify schema elements (tables, columns, and indexes). They are available from context menus in the **DB Structure view** of connected OpenEdge databases.

The **Edit Schema** wizards generate and execute SQL scripts that commit schema changes. You can view Edit Schema scripts in an editor and control the script's execution by using options available in the **Schema Edit Preferences** page. The two options that apply are Open script in Editor upon completion of wizard and Execute and commits schema change upon completion of wizard.

**OpenEdge Table wizard**
The **Table** wizard allows you to specify table properties for OpenEdge databases. You use the wizard to specify the properties of a new table or to change the properties of an existing table.

Start the **Table** wizard by clicking **Add Table** or **Edit Table**. Both are available from context menus in the **DB Structure view** and from the OpenEdge Table tab of the **DB Details view**.

The **Table** wizard allows you to specify or change the following properties:
<table>
<thead>
<tr>
<th>Table Name</th>
<th>Specifies a table name that is unique within the database. Table names must follow these conventions:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• A table name can be up to 32 characters long.</td>
</tr>
<tr>
<td></td>
<td>• A table name can consist of alphabetic characters (A-Z or a-z), digits (0-9), and the pound sign (#), dollar sign ($), percent sign (%), ampersand (&amp;), hyphen (-), and underscore (_). In addition, names must begin with a character (A-Z or a-z).</td>
</tr>
<tr>
<td></td>
<td>• You cannot use reserved ABL keywords as names.</td>
</tr>
<tr>
<td></td>
<td>• Names are not case-sensitive; they can be uppercase, lowercase, or a combination of both.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Partitioned</th>
<th>Creates a partitioned table without any partition policies defined on it.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Note:</strong> The Partitioned option is enabled only if the selected database is enabled for table partitioning. It is disabled if the Multi-tenant option is selected, since a table cannot be both multi-tenant and partitioned.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Multi-tenant</th>
<th>Creates a multi-tenant table.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Support default tenant</th>
<th>Allows you to associate a multi-tenant table with a storage area.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Area</th>
<th>Specifies the storage area from a list of storage areas defined for the database.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Note:</strong> The Area field is disabled if the Support default tenant check box is cleared while creating a multi-tenant table. The Area filed is also disabled when you select the Partitioned check box, since partition areas are defined while defining partition rules.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dump File</th>
<th>Specifies a unique filename for dumping table contents or definitions.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Label</th>
<th>Specifies the label that is used in error messages. This field is followed by a Field String Attribute field for controlling text preferences and space allocation.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>Describes the contents or purpose of the table for documentation purposes.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Validation</th>
<th>Defines what criteria to apply before row deletion is allowed. For example, in a customer table, to restrict deleting customers to those who have no outstanding orders, NOT (CAN-FIND(FIRST order OF customer))</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Message</th>
<th>Specifies an error message that is displayed when a record deletion is invalid. This field is followed by a Field String Attribute field for controlling text preferences and space allocation.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Category</th>
<th>Specifies the category of the OpenEdge tables to be displayed in the DB structure view. You can specify a category from the existing list or create a new one.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Hidden</th>
<th>Creates hidden tables.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Keep wizard open after adding table</th>
<th>Keeps the wizard open to add one or more other tables.</th>
</tr>
</thead>
</table>
See also

- Viewing and maintaining OpenEdge schema on page 932
- Working with OpenEdge database tables on page 932
- Connecting to a database on page 913
- Working with multi-tenant tables on page 934
- Columns wizard on page 970
- Index wizard on page 974
- OpenEdge DB Structure view on page 967

OpenEdge Columns wizard

The Columns wizard allows you to specify column properties for OpenEdge databases. You use it to specify the properties of a new column or to change the properties of an existing column.

Start the Columns wizard by clicking Add Column or Edit Column. Both are available from context menus in the DB Structure view and from the OpenEdge Table tab of the DB Details view.

The Columns wizard allows you to specify the following properties:

| Name | A column name that is unique within the selected table. A column name must start with a letter and can contain only letters, digits, or the following symbols: #, $, %, -, _, or &.
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong> The Name field is disabled while editing a partition key column since a partition key column cannot be renamed.</td>
<td></td>
</tr>
<tr>
<td>Data Type</td>
<td>A valid OpenEdge data type. The data types include CHARACTER, DATE, DECIMAL, INTEGER, INT64, LOGICAL, DATETIME, DATETIME-TZ, RAW, RECID, BLOB, and CLOB. The properties available in the Columns wizard change based on the choice of data type. You cannot change the data type of an existing column except to change from integer to INT64.</td>
</tr>
<tr>
<td><strong>Note:</strong> Changing the data type from INTEGER to INT64 is irreversible. The Data Type field is disabled while editing a column since the data type of a column cannot be changed.</td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>Maximum size (SQLWidth property) for the column. You can change the size of an existing column for these data types only: CHARACTER, DECIMAL, CLOB, BLOB, and RAW.</td>
</tr>
<tr>
<td>Scale</td>
<td>(Available only if Data Type is DECIMAL.) Maximum number of digits to the right of the decimal point. You cannot change the scale of an existing column. The scale defaults to the number of integers defined to the right of the decimal point in the format.</td>
</tr>
</tbody>
</table>
| Format | Display format for data in the column. You can choose character, date, logical, and numeric format options to determine how data is displayed on screens or in printed reports. This field is followed by a Field String Attribute field for controlling text preferences and space allocation. Each data type has a default format defined in the DB Navigator preferences on page 982.
| Label          | The column heading that appears on screens or in printed reports to identify the column. The Unknown value (?) causes the string that appears in Name to be used. This field is followed by a **Field String Attribute** field for controlling text preferences and space allocation.  
**Note:** If you use ? as a label, precede it with a backslash. For example: \\?. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Col Label</td>
<td>A heading that appears when a program lists data in columns. ? causes the string that appears in Label to be used. This field is followed by a <strong>Field String Attribute</strong> field for controlling text preferences and space allocation.</td>
</tr>
<tr>
<td>Init Value</td>
<td>The initial value, based on data type, for the column. This field is followed by a <strong>Field String Attribute</strong> field for controlling text preferences and space allocation.</td>
</tr>
</tbody>
</table>
| Order          | A number that indicates position of this column in relation to other columns in the table.  
**Note:** This order is used only when you display schema in an ABL client session. The order is not reflected in the **DB Structure** view, which displays columns in order of creation.  
By default, the order number for the first column is 10. Order numbers for subsequent columns are automatically incremented by 10. You can, however, modify the default order number to reposition the column. For example, if you want to position a new column between the first column (order number 10) and the second column (order number 20), you can set the order number to 15 for the new column. |
| Extent         | When selected, creates an array field and allows you to specify the number of elements. |
| Description    | A description of the contents or purpose of the column for documentation purposes. |
| Help Text      | A help message that appears whenever a user is prompted to input data. This field is followed by a **Field String Attribute** field for controlling text preferences and space allocation. |
| View As        | Specifies a UI widget (fill-in field, combo box, etc.) that represents the column on the screen. The widget you can use depends on the data type. See the VIEW-AS phrase in OpenEdge Development: ABL Reference for more information.  
**Note:** You can find OpenEdge manuals in the **Product Documentation** section of the Progress Software Developer's Network Web site. |
| Validation     | An expression that defines valid data for the column (for example custnum>0). |
| Val Msg        | The error message that is displayed when the user attempts to enter invalid data. This field is followed by a **Field String Attribute** field for controlling text preferences and space allocation. |
| Case sensitive | When selected, defines character columns as case-sensitive. |
When selected, makes the column mandatory. That is, it cannot have the Unknown value (?)..

(For BLOB or CLOB data types only) A value that indicates the maximum size for the column. You must indicate whether the value expresses Bytes, KiloBytes, MegaBytes, or GigaBytes.

(For CLOB data types only) Specifies a code page. A code page is a table that assigns a numeric value to each element in a collection of letters, numbers, punctuation, control codes, and other characters. The assignment is one-to-one: no two characters are assigned the same numeric value, and no two numeric values are assigned the same character.

(For CLOB data types only) Specifies collation preference. A collation is a set of rules that determine how character data is ordered. OpenEdge provides a set of collation tables to sort and compare character data.

Controls text preferences and how much space the ABL compiler allocates for a string within an r-code text segment.

The **Columns** wizard also has a tree view that shows the existing columns in the currently selected table. The following buttons are below the tree view:

- **Copy**
  - Allows one or more columns to be created from other columns in any table. Upon selecting a column and then selecting **OK**, the column information is copied into the **Column** tab folder view and the node is added to the tree view. If a column by the same name already exists in the table, a message is displayed and the column is renamed to be unique.

- **Remove**
  - Removes selected columns from the tree view.
  - **Note:** The **Remove** option is disabled while editing a partition key column, since a partition key cannot be removed.

- **Add**
  - Adds a new column node to the tree view and resets the fields in the **Columns** wizard to their initial values.

After one column is added, you can select **Finish**. This action constructs a SQL string, executes the code, and commits it. You can edit and save the SQL string by selecting **Open script in editor** upon completion of wizard on the **OpenEdge Schema Editing preferences page**.

**See also**
- Viewing and maintaining OpenEdge database schema on page 932
- Working with OpenEdge columns on page 940
- Connecting to a database on page 913
- Table wizard on page 968
- Index wizard on page 974

**Character display format examples**

Character display format examples
The following table shows examples of the different character display formats.

<table>
<thead>
<tr>
<th>Format options</th>
<th>Data</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>xxxxxxxx</td>
<td>These are characters</td>
<td>These are</td>
</tr>
<tr>
<td>x(9)</td>
<td>These are characters</td>
<td>These are</td>
</tr>
<tr>
<td>x(20)</td>
<td>These are characters</td>
<td>These are</td>
</tr>
<tr>
<td>xxx</td>
<td>These are characters</td>
<td>The</td>
</tr>
<tr>
<td>AAA-9999</td>
<td>abc1234</td>
<td>abc-1234</td>
</tr>
<tr>
<td>(999) 999-9999</td>
<td>6172754500</td>
<td>(617) 275-4500</td>
</tr>
<tr>
<td>!!</td>
<td>ma</td>
<td>MA</td>
</tr>
</tbody>
</table>

Trailing spaces are truncated in character fields. If a character field contains only one space, the space is truncated to a null value. You can use the TRIM function to truncate leading and trailing spaces.

**Date display format examples**

Date display format examples

The following table shows examples of the different date display formats.

<table>
<thead>
<tr>
<th>Format options</th>
<th>Data</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>99/99/99</td>
<td>8/10/2007</td>
<td>08/10/07</td>
</tr>
<tr>
<td>99-99-99</td>
<td>08/10/2007</td>
<td>08-10-07</td>
</tr>
<tr>
<td>99-99-99</td>
<td>08/10/2007</td>
<td>????????</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: The value 8/10/2007 exceeds the display format.</td>
</tr>
<tr>
<td>9999999</td>
<td>08/10/2007</td>
<td>081007</td>
</tr>
</tbody>
</table>
Logical display format examples

Logical display format examples

The following table shows examples of the different logical display formats.

<table>
<thead>
<tr>
<th>Format options</th>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes/no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Yes/No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>true/false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>shipped/waiting</td>
<td>shipped</td>
<td>waiting</td>
</tr>
<tr>
<td>male/female</td>
<td>male</td>
<td>female</td>
</tr>
</tbody>
</table>

OpenEdge Index wizard

The Index wizard allows you to specify index properties for OpenEdge databases. You use it to specify the properties of a new index or to change the properties of an existing index.

Start the Index wizard by clicking Add Index or Edit Index. Both are available from context menus in the DB Structure view and from the OpenEdge Index tab of the DB Details view.

The Index wizard allows you to specify the following properties:

<table>
<thead>
<tr>
<th>Name</th>
<th>An index name that is unique within the selected table. Index names must follow certain conventions.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scope</td>
<td>Defines whether the current index is Local or Global.</td>
</tr>
</tbody>
</table>

**Note:** Local indexes are created only for partitioned tables. A local index is a partition-aligned index that has a partition key column as the leading prefix of its constraint columns. By default, a global index is created for tables that are not partitioned; hence, the Local and Global options are disabled for tables that are not partitioned.
<table>
<thead>
<tr>
<th><strong>Area</strong></th>
<th>A storage area from a list of storage areas defined for the database.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong> When you select Local, the Area field is disabled, since partition areas are defined while defining partition rules.</td>
<td></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>A description of the index for documentation purposes.</td>
</tr>
<tr>
<td><strong>Primary</strong></td>
<td>When selected, designates the index as the most frequently used index. OpenEdge allows you to set one index as primary and uses it by default when retrieving or ordering records.</td>
</tr>
<tr>
<td><strong>Active</strong></td>
<td>When selected, enables the index to be used to retrieve or order records. An active index updates itself every time a new record is created, deleted, or modified.</td>
</tr>
<tr>
<td><strong>Unique</strong></td>
<td>When selected, indicates that every index key must be unique (for example, a social security number).</td>
</tr>
<tr>
<td><strong>Word Index</strong></td>
<td>When selected, indicates that all words in the field are index entries. An index that contains all the words from a text field (or array of text fields) allows you to search for records containing specific words or phrases.</td>
</tr>
<tr>
<td><strong>Note:</strong> A word index cannot be a primary index. Also, a local index cannot be a word index; so, when you select Local, the Word index option is disabled.</td>
<td></td>
</tr>
<tr>
<td><strong>Fields</strong></td>
<td>(Add Index only) A list of table fields and index fields that allows you to add or remove fields from the index.</td>
</tr>
<tr>
<td><strong>Note:</strong> While selecting fields to be added to a local index of a partitioned table, the list of fields must include all the fields defined in the partition policy details and in the same order.</td>
<td></td>
</tr>
<tr>
<td><strong>Descending</strong></td>
<td>(Add Index only) When selected, rows are sorted in decreasing order.</td>
</tr>
<tr>
<td><strong>Ascending</strong></td>
<td>(Add Index only) When selected, rows are sorted in increasing order. (This option is selected by default.)</td>
</tr>
</tbody>
</table>

**See also**
- SQL terminology on page 965
- Viewing and maintaining OpenEdge database schema on page 932
- Indexes on page 942
- Connecting to a database on page 913
- Table wizard on page 968
- Columns wizard on page 970
OpenEdge Add/Edit Sequence wizard

The Add/Edit Sequence wizard allows you to set the properties for new ABL sequences or to modify the properties of existing ABL sequences in an OpenEdge database.

Display the Add/Edit Sequence wizard from a context menu in the DB Structure view.

The Add/Edit Sequence wizard allows you to set the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Specifies the name of a new or existing sequence. Sequence names must follow certain conventions.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> You cannot edit the name of a multi-tenant sequence once it is created.</td>
</tr>
<tr>
<td>Initial Value</td>
<td>Specifies the first integer in the sequence. The default initial value is zero.</td>
</tr>
<tr>
<td>Increment by</td>
<td>Specifies a positive or negative integer value that changes the sequence value at each iteration. When you enter a negative value, the Upper Limit label changes to Lower Limit.</td>
</tr>
<tr>
<td>Upper/Lower Limit</td>
<td>Specifies an integer value that defines the upper or lower limit of the sequence, or the default value of unknown (?). When you specify a positive value for the Increment by field, you can define only an upper limit. If you specify a negative value, you can specify only a lower limit. You cannot specify an upper limit that is less than the initial value. Likewise, you cannot specify a lower limit that is greater than the initial value. If you accept the default, the maximum integer value on your system is used as the defined upper limit and the minimum integer value is used as the defined lower limit.</td>
</tr>
<tr>
<td>Cycle at limit</td>
<td>Specifies a value that defines the limit of the sequence. When the sequence reaches this value, it starts again at the value defined in Initial Value.</td>
</tr>
<tr>
<td>Multi-tenant</td>
<td>Creates a multi-tenant sequence.</td>
</tr>
</tbody>
</table>

**Note:** For more information about sequences, see the ABL manuals in the OpenEdge Product Documentation section of the Progress Communities Web site.

See also

- Working with OpenEdge database sequences on page 944
- Working with multi-tenant sequences on page 946

OpenEdge Create/Modify Trigger wizard

The Create/Modify Trigger wizard allows you to set the properties for new ABL triggers or modify the properties of existing ABL triggers in an OpenEdge database.
Display the Create/Modify Trigger wizard from a context menu in the DB Structure view.

**Note:** To use the Create/Modify Trigger wizard, there must be at least one OpenEdge project opened. If no project is open, an error message appears.

The Create/Modify Trigger wizard allows you to set the following properties:

<table>
<thead>
<tr>
<th>Table</th>
<th>A valid table name from a list of tables in the PUB schema.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column Trigger</td>
<td>When the option is selected, displays all columns for the selected table in the Column combo box. The Event field is set to ASSIGN (since that is the only event for a column trigger).</td>
</tr>
<tr>
<td>Column</td>
<td>A list of the columns in the selected table.</td>
</tr>
<tr>
<td>Event</td>
<td>A list of available trigger types including: CREATE, DELETE, WRITE, FIND, REPLICATION-CREATE, REPLICATION-DELETE, REPLICATION-WRITE, and ASSIGN (for column triggers).</td>
</tr>
<tr>
<td>Create new trigger file</td>
<td>When the option is selected, the system creates a new file by the system in a specified project and folder. You specify the project and folder in the New File dialog, which appears when you select the Browse button. When you specify the project and folder, they appear in the read-only Parent Folder field. The read-only Propath field displays the PROPATH setting for the selected OpenEdge project.</td>
</tr>
<tr>
<td>Attach existing trigger</td>
<td>When the option is selected, the system attaches a specified trigger to this trigger. You select an existing trigger in the Trigger Selection dialog box, which appears when you select the Browse button. After you select a trigger, the trigger file name appears in the read-only File Name field.</td>
</tr>
<tr>
<td>Override</td>
<td>When the option is selected, users can redefine and overwrite triggers at run time.</td>
</tr>
<tr>
<td>Check CRC</td>
<td>When the option is selected, the system checks the validity of the trigger at run time. CRC (Cyclical Redundancy Check) is an algorithm that verifies the integrity of executables. It compares a stored number in the executable with a number calculated from the schema. If the numbers do not match (after a schema change, for example), the executable must be recompiled.</td>
</tr>
</tbody>
</table>

After you complete the Create/Modify Trigger wizard, the ABL Editor opens.

**Note:** For more information about triggers, see the ABL manuals in the OpenEdge Product Documentation section of the Progress Communities Web site.

**See also**

Working with OpenEdge database triggers on page 948

**Field string attributes**

Field string attributes control text preferences and how much space the ABL compiler allocates for a string within an r-code text segment.
The following table describes the string attribute control characters you can enter in the **Field String Attributes** fields of the **Edit Schema wizards**:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Center the string.</td>
</tr>
<tr>
<td>L</td>
<td>Left-justify the string.</td>
</tr>
<tr>
<td>nnn</td>
<td>The maximum amount of space allocated for the text segment. (Specify a number from 1 to 999.) If you do not specify a value, the default number of characters for the field type is used.</td>
</tr>
<tr>
<td>R</td>
<td>Right-justify the string.</td>
</tr>
<tr>
<td>T</td>
<td>Remove trailing spaces. (Spaces are not physically removed from the string, but are not displayed.)</td>
</tr>
<tr>
<td>U</td>
<td>Specify that the Translation Manager should not translate the string.</td>
</tr>
</tbody>
</table>

You can specify the characters in any order but use each character only once. You can use only one of the text justification codes (L, C, or R). If you do not specify justification, the ABL Compiler left-justifies the string by default. The compiler uses these codes to justify strings within the allocated bytes in the r-code text segment, not on the screen.

**See also**
- Table wizard on page 968
- Columns wizard on page 970
- Index wizard on page 974

**Oracle database**

You can view Oracle database schema by using the a Details view and a Structure view.

**Oracle DB Details view**

The **DB Details** view is a read-only display of information specific to the selected node in the **DB Structure** view.

**Note:** This detailed view is available only for certain nodes. If the detailed view is unavailable for a node you select in the **DB Structure** view, no information appears in the **DB Details** view.

The content of the **DB Details** view changes depending on which node you select in the DB Structure view. There are four top-level nodes: Database, Monitor, Security, and Instance.

**Database**

When you select a child of one of the nodes listed in the left column of the following table, the corresponding information appears in the **DB Details** view:
<table>
<thead>
<tr>
<th>TABLE</th>
<th>For a child node, displays details about the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Columns - A list of all columns and related SQL information.</td>
</tr>
<tr>
<td></td>
<td>• Indexes - A list of indexes in the selected table.</td>
</tr>
<tr>
<td></td>
<td>• Primary key - A list of the primary keys defined for the selected table.</td>
</tr>
<tr>
<td></td>
<td>• Foreign key - A list of any foreign key restraints defined for a selected SQL table.</td>
</tr>
<tr>
<td></td>
<td>• Preview - A subset of the records in the selected table. You can set the number of rows returned in the Preferences page.</td>
</tr>
<tr>
<td></td>
<td>• Row count - The total number of rows in the selected table.</td>
</tr>
<tr>
<td></td>
<td>• Privileges - A list of both table and column privileges (as a toggle).</td>
</tr>
<tr>
<td></td>
<td>• Dependent Objects - A list of objects dependent on that table, identified by owner, type, and name.</td>
</tr>
<tr>
<td></td>
<td>• Status - Status details, including properties and their values.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VIEW</th>
<th>For a child node, displays details about the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• Columns - A list of all columns and related SQL information.</td>
</tr>
<tr>
<td></td>
<td>• Indexes - A list of indexes in the selected table.</td>
</tr>
<tr>
<td></td>
<td>• Primary key - A list of the primary keys defined for the selected table.</td>
</tr>
<tr>
<td></td>
<td>• Foreign key - A list of any foreign key restraints defined for a selected SQL table.</td>
</tr>
<tr>
<td></td>
<td>• Preview - A subset of the records in the selected table. You can set the number of rows returned in the Preferences page.</td>
</tr>
<tr>
<td></td>
<td>• Row count - The total number of rows in the selected table.</td>
</tr>
<tr>
<td></td>
<td>• Privileges - A list of both table and column privileges (as a toggle).</td>
</tr>
<tr>
<td></td>
<td>• Dependent Objects - A list of objects dependent on that table, identified by owner, type, and name.</td>
</tr>
<tr>
<td></td>
<td>• Status - Status details, including properties and their values.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SEQUENCE</th>
<th>Displays the properties of the sequence, including its current value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>PACKAGE</td>
<td>Displays details about each package, including the SQL code, properties and their values, and a list of dependent objects.</td>
</tr>
<tr>
<td>PACKAGE BODY</td>
<td>Displays details about each package body, including the SQL code, and properties and their values.</td>
</tr>
<tr>
<td>FUNCTION</td>
<td>Displays details about each function, including parameters, SQL code, properties and their values, and a list of dependent objects.</td>
</tr>
<tr>
<td>PROCEDURE</td>
<td>Displays details about a procedure, including parameters, SQL code, properties and their values, and a list of dependent objects.</td>
</tr>
<tr>
<td>TRIGGER</td>
<td>Displays details about a trigger, including properties and their values, details about other properties, and SQL code.</td>
</tr>
</tbody>
</table>
SECTIONS | Displays details about the current schema. Includes the columns Username, OS User, Session ID, Serial No, Process ID, LockWait, Status, PGA (Kb), UGA (Kb), Module, Machine, Program, and Logon Time.

JAVA SOURCE | Displays details about a Java object, including information about properties and a display of the Java object source.

DATABASE LINK | Displays details about a database link, including the user name, host, and when the link was created.

Monitor

When you select a child of one of the nodes listed in the left column of the following table, the corresponding information appears in the DB Details view:

| SESSIONS | Displays the following session information: Username, OS User, Session ID, Serial No, Process ID, LockWait, Status, PGA (Kb), UGA (Kb), Module, Machine, Program, and Logon Time.
| JOBS | Displays the following job information: Log User, Priv User, Schema User, Last Date, This Date, Next Date, Total Time, Broken, Interval, Failures, What, NLS Env, and Misc Env.
| TOP SQL | Displays the following SQL executions data: SQL Text, Reads/Execution, Buffer Gets, Disk Reads, Executions, Sorts, Address, and Hash Value.

Security

When you select a child of the Users node, a list of all authenticated users, identified by UserName, UserID, and when created, appears in the DB Details view.

Instance

When you select a child of one of the nodes listed in the left column of the following table, the corresponding information appears in the DB Details view:

| System Parameters | Displays all the system parameters, which are retrieved from the v$system_parameter table. Includes the columns NUM, NAME, TYPE, and VALUE.
| TableSpace | Displays child nodes for every existing table space. Includes Allocation Type, Contents, Extent Management, Initial Extent, Logging, Max Extent, Min ExtLen, Min Extent, Next Extent, PCT Increase, and Status.

Oracle DB Structure View

When you open a connection in the Connections view, the schema for the database appears in the DB Structure view. The DB Structure view is a tree view that shows a separate structure for each schema under the root node Database.

Note: You can configure DB Navigator to display or hide schemas. By default, catalogs are displayed while schemas are hidden from view. See DB Navigator preferences on page 982 for more information.

When you select certain nodes in the DB Structure view, properties and values appear in the DB Details view.
The top-level nodes in the **Oracle DB Structure** view are Database, Monitor, Security, and Instance. You can use **filters** to show only certain schema items in the view; you cannot, however, create or edit schema.

**Database**

The root node **Database** is at the top level of the **DB Structure** view. If the database supports catalogs and you have selected **Show catalog** in the DB Navigator preferences, all catalogs appear below the database node. If you have selected **Show schema** in the DB Navigator preferences, all schemas appear below the database node.

Beneath the schema and catalog nodes, the following child nodes appear:

<table>
<thead>
<tr>
<th>SYNONYM</th>
<th>All aliases that were created in SQL against tables. SQL statements can use these aliases instead of the name specified when the schema item, such as table, view, or synonym, for example, was created. The child nodes contain only tables, views, or other synonyms. They do not display columns or indexes of the tables.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE</td>
<td>Displays all application tables. Table nodes expand to show columns and indexes. When you select a table name, data and meta-data appear in the <strong>DB Details view</strong>.</td>
</tr>
<tr>
<td>VIEW</td>
<td>Displays the names of all created views. A SQL view allows you to extract rows and columns from one or more related tables and to treat the resulting data set as a virtual table that can be viewed or updated.</td>
</tr>
<tr>
<td>SEQUENCE</td>
<td>Displays all sequences, which create an incremented, serial list of numbers applied to a column in a database table.</td>
</tr>
<tr>
<td>PACKAGE</td>
<td>Displays all packages, which contain specific procedures, functions, and other schema items in one entity.</td>
</tr>
<tr>
<td>PACKAGE BODY</td>
<td>Displays details about each package body, including the SQL code, and properties and their values.</td>
</tr>
<tr>
<td>FUNCTION</td>
<td>Displays all functions, or pieces of a program that you can use separately from the remainder of the program.</td>
</tr>
<tr>
<td>PROCEDURE</td>
<td>Displays all procedures that are stored in the database and available for reuse.</td>
</tr>
<tr>
<td>TRIGGER</td>
<td>Displays all triggers, which you define to run when you modify a table or view or when a particular action occurs.</td>
</tr>
<tr>
<td>SESSIONS</td>
<td>Displays information about the current user-to-database connection in the <strong>DB Details view</strong>.</td>
</tr>
<tr>
<td>JAVA SOURCE</td>
<td>Displays all Java objects.</td>
</tr>
<tr>
<td>DATABASE LINK</td>
<td>Displays all database links. Each database link is present as a local database object and contains specifics necessary for connecting to another database.</td>
</tr>
</tbody>
</table>

**Monitor**

When you select the **Monitor** node, the following subnodes appear:
<table>
<thead>
<tr>
<th>SECTIONS</th>
<th>Displays information about the current session, such as username, Session ID, machine name, and the user's logon time, in the <strong>DB Details view</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOBS</td>
<td>Displays information about a job, or task, that you run once or on a particular schedule, in the <strong>DB Details view</strong>.</td>
</tr>
<tr>
<td>TOP SQL</td>
<td>Displays SQL executions in the <strong>DB Details view</strong>, enabling you to see how resources are affected when a SQL statement is executing.</td>
</tr>
</tbody>
</table>

**Security**

When you select the **Security** node, the following subnode appears:

| Users | Displays all authenticated users by username, user ID, and the date on which the user was created. |

**Instance**

When you select the **Instance** node, the following subnodes appear:

<table>
<thead>
<tr>
<th>System Parameters</th>
<th>Displays all system parameters (retrieved from the v$system_parameter table) in a viewer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TableSpace</td>
<td>Displays all table spaces, or logical storage units, in the database.</td>
</tr>
</tbody>
</table>

**See also**

- Oracle **DB Details view** on page 978
- DB Navigator preferences on page 982

**Preferences**

You can access the preference pages for DB Navigator by choosing **Windows > Preferences > Progress Databases > DB Navigator**.

**DB Navigator preferences**

The following list describes the preferences you can set in the **DB Navigator Preferences** page:

<table>
<thead>
<tr>
<th>Preference</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preview row limit</td>
<td>Specifies the number of rows to return in the Preview tab folder. The recommended range is 1-100. The default is 80.</td>
</tr>
<tr>
<td>SQL Results row limit</td>
<td>Specifies the number of rows to return after execution of SQL Select statements. The recommended range is 100-5000. The default is 2000.</td>
</tr>
<tr>
<td>SQL Results rows to batch</td>
<td>Specifies the number of rows to batch when retrieving records. The default is 100.</td>
</tr>
<tr>
<td>SQL Editor separator</td>
<td>Specifies the separator to use for separating multiple SQL statements. The default is a semicolon.</td>
</tr>
<tr>
<td>Setting</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Auto commit</td>
<td>Determines the default AutoCommit mode for a connection. If you select the <strong>Auto commit</strong> option in the DB Navigator preferences, AutoCommit is selected by default for each SQL connection profile. <em>(You can, however, select or deselect <strong>AutoCommit</strong> for any connection profile individually.)</em> When you select <strong>AutoCommit</strong>, all executed SQL statements that update the database are automatically committed. If you do not select <strong>AutoCommit</strong>, you must manually commit all transactions by either clicking the <strong>Commit</strong> button in the Connections view or issuing a COMMIT statement. <strong>Note:</strong> You see the <strong>Auto commit</strong> option only if you do not have the <strong>Auto-Login on connect</strong> option selected for the connection profile.</td>
</tr>
<tr>
<td>Commit On Close</td>
<td>Specifies that commit is performed on closing of the connection. You can enable this option only if the <strong>AutoCommit</strong> option is not selected.</td>
</tr>
<tr>
<td>Columns auto-completing assistance</td>
<td>Enables auto-completion for columns within the SQL Editor. If this option is not selected, columns are available in the auto-completion pop-up only if the column's table node has been expanded. Be aware that auto-completion can slow performance during connection.</td>
</tr>
<tr>
<td>Save and restore last selected node in Database Structure View</td>
<td>Saves the last selected node in the <strong>DB Structure</strong> view, then expands and selects the saved node when the connection is restarted.</td>
</tr>
<tr>
<td>Append table name to column</td>
<td>Causes the table name to be appended when column names are pasted to the clipboard (for example, Customer.Name).</td>
</tr>
<tr>
<td>Enclose in quotes</td>
<td>Causes quotes to be added to column names, table names, or schema when copied to the clipboard (for example, &quot;CustNum&quot;, &quot;Name&quot;) when generating SQL select statements or creating a table script. <strong>Note:</strong> Column names that contain leading underscores (for example, _Buffer-ID) must be enclosed by quotes. Otherwise, the SQL Editor does not recognize them.</td>
</tr>
<tr>
<td>Delimiter</td>
<td>Specifies the character used as a delimiter when multiple objects are copied to the clipboard.</td>
</tr>
<tr>
<td>Show catalog or Show schema</td>
<td>If <strong>Show catalog</strong> is selected and the database supports catalogs, then all catalogs are listed beneath the root node and all objects per catalog (tables, views, etc.) are grouped together. If <strong>Show schema</strong> is selected, all of the schemas are listed beneath the root node. <strong>Note:</strong> When you change from one preference to the other, you must refresh the DB Structure view, or close and reopen the connection.</td>
</tr>
</tbody>
</table>

**See also**

[Setting preferences](#) on page 910
SQL Connection profile preferences

The Connection Profiles preferences page displays SQL connection profiles. This page includes the following controls:

<table>
<thead>
<tr>
<th>Add</th>
<th>Adds a new SQL connection profile for the workspace.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit</td>
<td>Modifies aspects of an existing SQL connection profile.</td>
</tr>
<tr>
<td>Copy</td>
<td>Copies an existing SQL connection profile.</td>
</tr>
<tr>
<td>Remove</td>
<td>Deletes a SQL connection profile.</td>
</tr>
</tbody>
</table>

See also

Setting preferences on page 910
OpenEdge Schema Editing preferences on page 985
SQL Editor preferences on page 986

JDBC Drivers preferences

The JDBC Drivers preferences page allows you to configure JDBC drivers. The page provides a list of available JDBC drivers currently defined for the workspace. Using this page, you can:

<table>
<thead>
<tr>
<th>Add</th>
<th>Add a new JDBC driver. The Add new driver dialog allows you to add a new JDBC driver for the workspace. The page displays Java Class Paths and Extra Class Paths already defined for the workspace. Use the List Drivers button to display existing drivers. Use the drop-down menu to select the Driver Class Name. When adding a JDBC driver, you must specify the name of the driver. Type a URL for the driver in the Example URL field. If you are using an IPv6 address in the URL for OpenEdge or MS SQL server databases, be sure to insert a bracket before and after the address; for example, [thehostmachine].</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit</td>
<td>Modify aspects of an existing JDBC driver. Select the driver you want to edit from the list of available drivers, then click Edit.</td>
</tr>
<tr>
<td>Copy</td>
<td>Copy a driver. Select the driver you want to copy from the list of available drivers, then click Copy.</td>
</tr>
</tbody>
</table>

Note: See the documentation for the driver if you require more information about the driver.
Delete a JDBC driver. Select the driver you want to delete from the list of available drivers, then click **Remove**.

Set the selected JDBC driver as the default.

### See also
- **Setting preferences** on page 910
- **Configuring a JDBC driver** on page 912

### OpenEdge Schema Editing preferences

The following list describes the preferences you can set in the **Databases > OpenEdge Schema Editing Preferences** page:

<table>
<thead>
<tr>
<th>Preference</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open script in editor upon completion of wizard</td>
<td>When the option is enabled, a script that is generated by one of the Edit Schema wizards opens in the SQL editor.</td>
</tr>
</tbody>
</table>
| Execute and commit Schema change on completion of wizard | When the option is enabled, a SQL script generated by one of the Edit Schema wizards executes, is committed, and opens in the SQL editor.  
When the option is disabled, the SQL script does not update the schema. However, the script appears in the SQL editor. This is useful when you want to create a script but do not want to change the schema. |

**Note**: This option can be disabled only if the **Open script in editor** option is selected.

<table>
<thead>
<tr>
<th>Preference</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grant public SQL rights</td>
<td>Allows users who do not have DBA privileges to view data or edit tables created by the Edit Schema wizard.</td>
</tr>
<tr>
<td>Default Character Size</td>
<td>Establishes the default size for new character fields.</td>
</tr>
<tr>
<td>Default Decimal Size</td>
<td>Establishes the default size for new decimal fields.</td>
</tr>
<tr>
<td>Update format to reflect entered size value</td>
<td>When this option is enabled, you can enter a value in the <strong>Size to Format ratio</strong> field. This value is used when entering the size value for a character field. The value of the format field is updated to reflect the entered size as you type.</td>
</tr>
<tr>
<td>Default Format</td>
<td>Displays default formats for all ABL data types. You can change the defaults by selecting a data type and clicking the <strong>Set</strong> button.</td>
</tr>
</tbody>
</table>

### See also
- **Setting preferences** on page 910
- **Edit Schema wizards** on page 968
- **SQL Editor Preferences** on page 986
- **SQL Connection profile preferences** on page 984
SQL Editor preferences

The SQL Editor Preferences page allows you to define the SQL editor font and color coding for various elements such as tables, columns, and SQL key words. The preferences also allow you to specify various Export to Clipboard options.

See also

SQL scripting on page 916
SQL Editor on page 961
SQL Connection profile preferences on page 984