OpenEdge Architect Online Help

This PDF contains the same information as the OpenEdge Architect online help. When using OpenEdge Architect, the online help is the recommended method for accessing this information. You can access the online help from the OpenEdge Architect Help menu.
Copyright

© 2009 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.

Actional, Apama, Apama (and Design), Artix, Business Empowerment, DataDirect (and design), DataDirect Connect, DataDirect Connect64, DataDirect Technologies, DataDirect XML Converters, DataDirect XQuery, DataXtend, Dynamic Routing Architecture, EdgeXtend, Empowerment Center, Fathom, IntelliStream, IONA, IONA (and design), Making Software Work Together, Mindreef, ObjectStore, OpenEdge, Orbix, PeerDirect, POSSENET, Powered by Progress, PowerTier, Progress, Progress DataXtend, Progress Dynamics, Progress Business Empowerment, Progress Empowerment Center, Progress Empowerment Program, Progress OpenEdge, Progress Profiles, Progress Results, Progress Software Developers Network, Progress Sonic, ProVision, PS Select, SequeLink, Shadow, SOAPscope, SOAPStation, Sonic, Sonic ESB, SonicMQ, Sonic Orchestration Server, SonicSynergy, SpeedScript, Stylus Studio, Technical Empowerment, WebSpeed, Xcalia (and design), and Your Software, Our Technology–Experience the Connection are registered trademarks of Progress Software Corporation or one of its subsidiaries or affiliates in the U.S. and/or other countries. AccelEvent, Apama Dashboard Studio, Apama Event Manager, Apama Event Modeler, Apama Event Store, Apama Risk Firewall, AppsAlive, AppServer, ASPen, ASP-in-a-Box, BusinessEdge, Business Making Progress, Cache-Forward, DataDirect Spy, DataDirect SupportLink, Fuse, Fusion Mediation Router, Fuse Message Broker, Fuse Services Framework, Future Proof, GVAC, High Performance Integration, ObjectStore Inspector, ObjectStore Performance Expert, OpenAccess, Orbacus, Pantero, POSSE, ProDataSet, Progress ESP Event Manager, Progress ESP Event Modeler, Progress Event Engine, Progress RFID, Progress Software Business Making Progress, PSE Pro, SectorAlliance, SeeThinkAct, Shadow z/Services, Shadow z/Direct, Shadow z/Events, Shadow z/Presentation, Shadow Studio, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, Sonic Business Integration Suite, Sonic Process Manager, Sonic Collaboration Server, Sonic Continuous Availability Architecture, Sonic Database Service, Sonic Workbench, Sonic XML Server, StormGlass, The Brains Behind BAM, WebClient, Who Makes Progress, and Your World. Your SOA, are trademarks or service marks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and other countries. Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. Any other trademarks contained herein are the property of their respective owners.

Third Party Acknowledgements:

OpenEdge includes AdventNet - Agent Toolkit licensed from AdventNet, Inc. http://www.adventnet.com. All rights to such copyright material rest with AdventNet.

OpenEdge includes ANTLR (Another Tool for Language Recognition) software Copyright © 2003-2006, Terence Parr All rights reserved. Neither the name of the author nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. Software distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License agreement
that accompanies the product.

OpenEdge includes software developed by the Apache Software Foundation (http://www.apache.org/). Copyright © 1999 The Apache Software Foundation. All rights reserved (Xerces C++ Parser (XML) and Xerces2 Java Parser (XML)); Copyright © 1999-2002 The Apache Software Foundation. All rights reserved (Xerces Parser (XML); and Copyright © 2000-2003 The Apache Software Foundation. All rights reserved (Ant). The names “Apache,” “Xerces,” “ANT,” and “Apache Software Foundation” must not be used to endorse or promote products derived from this software without prior written permission. Products derived from this software may not be called “Apache”, nor may “Apache” appear in their name, without prior written permission of the Apache Software Foundation. For written permission, please contact apache@apache.org. Software distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License agreement that accompanies the product.

OpenEdge includes Concurrent Java software Copyright 1994-2000 Sun Microsystems, Inc. All Rights Reserved. -Neither the name of or trademarks of Sun may be used to endorse or promote products including or derived from the Java Software technology without specific prior written permission; and Redistributions of source or binary code must contain the above copyright notice, this notice and the following disclaimers: This software is provided "AS IS," without a warranty of any kind. ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE HEREBY EXCLUDED. SUN MICROSYSTEMS, INC. AND ITS LICENSORS SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THE SOFTWARE OR ITS DERIVATIVES. IN NO EVENT WILL SUN MICROSYSTEMS, INC. OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF THE USE OF OR INABILITY TO USE SOFTWARE, EVEN IF SUN MICROSYSTEMS, INC. HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

OpenEdge includes DataDirect software Copyright © 1991-2007 Progress Software Corporation and/or its subsidiaries or affiliates. All Rights Reserved. (DataDirect Connect for JDBC Type 4 driver); Copyright © 1993-2009 Progress Software Corporation and/or its subsidiaries or affiliates. All Rights Reserved. (DataDirect Connect for JDBC); Copyright © 1988-2007 Progress Software Corporation and/or its subsidiaries or affiliates. All Rights Reserved. (DataDirect Connect for ODBC); and Copyright © 1988-2007 Progress Software Corporation and/or its subsidiaries or affiliates. All Rights Reserved. (DataDirect Connect64 for ODBC).

OpenEdge includes DataDirect Connect for ODBC and DataDirect Connect64 for ODBC software, which include ICU software 1.8 and later - Copyright © 1995-2003 International Business Machines Corporation and others All rights reserved. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

OpenEdge includes DataDirect Connect for ODBC and DataDirect Connect64 for ODBC software, which include software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/). Copyright © 1998-2006 The OpenSSL
OpenEdge includes DataDirect products for the Microsoft SQL Server database which contain a licensed implementation of the Microsoft TDS Protocol.

OpenEdge includes software authored by David M. Gay. Copyright © 1991, 2000, 2001 by Lucent Technologies (dtoa.c); Copyright © 1991, 1996 by Lucent Technologies (g_fmt.c); and Copyright © 1991 by Lucent Technologies (rnd_prod.s). Permission to use, copy, modify, and distribute this software for any purpose without fee is hereby granted, provided that this entire notice is included in all copies of any software which is or includes a copy or modification of this software and in all copies of the supporting documentation for such software. THIS SOFTWARE IS BEING PROVIDED "AS IS", WITHOUT ANY EXPRESS OR IMPLIED WARRANTY. IN PARTICULAR, NEITHER THE AUTHOR NOR LUCENT MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND CONCERNING THE MERCHANTABILITY OF THIS SOFTWARE OR ITS FITNESS FOR ANY PARTICULAR PURPOSE.

OpenEdge includes software authored by David M. Gay. Copyright © 1998-2001 by Lucent Technologies All Rights Reserved (decstrtod.c; strtodg.c); Copyright © 1998, 2000 by Lucent Technologies All Rights Reserved (decstrtof.c; srtord.c); Copyright © 1998 by Lucent Technologies All Rights Reserved (dmisc.c; dtoa.h; gethex.c; gmisc.c; sum.c); Copyright © 1998, 1999 by Lucent Technologies All Rights Reserved (gdtoa.c; misc.c; smisc.c; ulp.c); Copyright © 1998-2000 by Lucent Technologies All Rights Reserved (gdtoaimp.h); Copyright © 2000 by Lucent Technologies All Rights Reserved (hd_init.c). Full copies of these licenses can be found in the installation directory, in the c:/OpenEdge/licenses folder. Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that the copyright notice and this permission notice and warranty disclaimer appear in supporting documentation, and that the name of Lucent or any of its entities not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. LUCENT DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL LUCENT OR ANY OF ITS ENTITIES BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

OpenEdge includes http package software developed by the World Wide Web Consortium. Copyright © 1994-2002 World Wide Web Consortium, (Massachusetts Institute of Technology, European Research Consortium for Informatics and Mathematics, Keio University). All rights reserved. This work is distributed under the W3C® Software License [http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231] in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

OpenEdge includes ICU software 1.8 and later - Copyright © 1995-2003 International Business Machines Corporation and others All rights reserved. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s)
and this permission notice appear in supporting documentation.


OpenEdge includes Infragistics NetAdvantage for .NET v2009 Vol 2 Copyright © 1996-2009 Infragistics, Inc. All rights reserved.

OpenEdge includes JSTL software Copyright 1994-2006 Sun Microsystems, Inc. All Rights Reserved. Software distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License agreement that accompanies the product.

OpenEdge includes OpenSSL software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/). Copyright © 1998-2007 The OpenSSL Project. All rights reserved. This product includes cryptographic software written by Eric Young (eay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com). Copyright © 1995-1998 Eric Young (eay@cryptsoft.com) All rights reserved. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact openssl-core@openssl.org. Products derived from this software may not be called "OpenSSL" nor may "OpenSSL" appear in their names without prior written permission of the OpenSSL Project. Software distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License agreement that accompanies the product.

OpenEdge includes Quartz Enterprise Job Scheduler software Copyright © 2001-2003 James House. All rights reserved. Software distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License agreement that accompanies the product. This product uses and includes within its distribution, software developed by the Apache Software Foundation (http://www.apache.org/).

OpenEdge includes code licensed from RSA Security, Inc. Some portions licensed from IBM are available at http://oss.software.ibm.com/icu4j/.

OpenEdge includes the RSA Data Security, Inc. MD5 Message-Digest Algorithm. Copyright ©1991-2, RSA Data Security, Inc. Created 1991. All rights reserved.

OpenEdge includes Sonic software, which includes software developed by Apache Software Foundation (http://www.apache.org/). Copyright © 1999-2000 The Apache Software Foundation. All rights reserved. The names “Ant”, “Axis”, “Xalan”, “FOP”, “The Jakarta Project”, “Tomcat”, “Xerces” and/or “Apache Software Foundation” must not be used to endorse or promote products derived from the Product without prior written permission. Any product derived from the Product may not be called “Apache”, nor may “Apache” appear in their name, without prior written permission. For written permission, please contact apache@apache.org.

OpenEdge includes Sonic software, which includes software Copyright © 1999 CERN - European Organization for Nuclear Research. Permission to use, copy, modify, distribute and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation. CERN makes no representations about the suitability of this software for any purpose. It is provided "as is" without expressed or implied warranty.
OpenEdge includes Sonic software, which includes software developed by ExoLab Project (http://www.exolab.org/). Copyright © 2000 Intalio Inc. All rights reserved. The names “Castor” and/or “ExoLab” must not be used to endorse or promote products derived from the Products without prior written permission. For written permission, please contact info@exolab.org. Exolab, Castor and Intalio are trademarks of Intalio Inc.

OpenEdge includes Sonic software, which includes software developed by IBM. Copyright © 1995-2003 International Business Machines Corporation and others. All rights reserved. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation. Software distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License agreement that accompanies the product. Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

OpenEdge includes Sonic software, which includes the JMX Technology from Sun Microsystems, Inc. Use and Distribution is subject to the Sun Community Source License available at http://sun.com/software/communitysource.

OpenEdge includes Sonic software, which includes software developed by the ModelObjects Group (http://www.modelobjects.com). Copyright © 2000-2001 ModelObjects Group. All rights reserved. The name “ModelObjects” must not be used to endorse or promote products derived from this software without prior written permission. Products derived from this software may not be called “ModelObjects”, nor may “ModelObjects” appear in their name, without prior written permission. For written permission, please contact djacobs@modelobjects.com.

OpenEdge includes Sonic software, which includes code licensed from Mort Bay Consulting Pty. Ltd. The Jetty Package is Copyright © 1998 Mort Bay Consulting Pty. Ltd. (Australia) and others.

OpenEdge includes Sonic software, which includes files that are subject to the Netscape Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.mozilla.org/NPL/. Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License. The Original Code is Mozilla Communicator client code, released March 31, 1998. The Initial Developer of the Original Code is Netscape Communications Corporation. Portions created by Netscape are Copyright 1998-1999 Netscape Communications Corporation. All Rights Reserved.

OpenEdge includes Sonic software, which includes software developed by the University Corporation for Advanced Internet Development http://www.ucaid.edu Internet2 Project. Copyright © 2002 University Corporation for Advanced Internet Development, Inc. All rights reserved. Neither the name of OpenSAML nor the names of its contributors, nor Internet2, nor the University Corporation for Advanced Internet Development, Inc., nor UCAID may be used to endorse or promote products derived from this software and products derived from this software may not be called OpenSAML, Internet2, UCAID, or the University Corporation for Advanced Internet Development, nor may OpenSAML appear in their name without prior written permission.
of the University Corporation for Advanced Internet Development. For written
permission, please contact opensaml@opensaml.org.

OpenEdge includes the UnixWare platform of Perl Runtime authored by Kiem-Phong Vo
and David Korn. Copyright © 1991, 1996 by AT&T Labs. Permission to use, copy, modify,
and distribute this software for any purpose without fee is hereby granted, provided that
this entire notice is included in all copies of any software which is or includes a copy or
modification of this software and in all copies of the supporting documentation for such
software. THIS SOFTWARE IS BEING PROVIDED "AS IS", WITHOUT ANY EXPRESS OR
IMPLIED WARRANTY. IN PARTICULAR, NEITHER THE AUTHORS NOR AT&T LABS MAKE
ANY REPRESENTATION OR WARRANTY OF ANY KIND CONCERNING THE
MERCHANTABILITY OF THIS SOFTWARE OR ITS FITNESS FOR ANY PARTICULAR
PURPOSE.

OpenEdge includes Vermont Views Terminal Handling Package software developed by

OpenEdge includes XML Tools, which includes versions 8.9 of the Saxon XSLT and
XQuery Processor from Saxonica Limited (http://www.saxonica.com/) which are
available from SourceForge (http://sourceforge.net/projects/saxon/). The Original Code
of Saxon comprises all those components which are not explicitly attributed to other
parties. The Initial Developer of the Original Code is Michael Kay. Until February 2001
Michael Kay was an employee of International Computers Limited (now part of Fujitsu
Limited), and original code developed during that time was released under this license
by permission from International Computers Limited. From February 2001 until February
2004 Michael Kay was an employee of Software AG, and code developed during that time
was released under this license by permission from Software AG, acting as a
"Contributor". Subsequent code has been developed by Saxonica Limited, of which
Michael Kay is a Director, again acting as a "Contributor". A small number of modules,
or enhancements to modules, have been developed by other individuals (either written
especially for Saxon, or incorporated into Saxon having initially been released as part of
another open source product). Such contributions are acknowledged individually in
comments attached to the relevant code modules. All Rights Reserved. The contents of
the Saxon files are subject to the Mozilla Public License Version 1.0 (the "License"); you
may not use these files except in compliance with the License. You may obtain a copy of
the License at http://www.mozilla.org/MPL/ and a copy of the license can also be found
in the installation directory, in the c:/OpenEdge/licenses folder. Software distributed
under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND,
either express or implied. See the License for the specific language governing rights and
limitations under the License.

OpenEdge includes XML Tools, which includes Xs3P v1.1.3. The contents of this file are
subject to the DSTC Public License (DPL) Version 1.1 (the "License"); you may not use
this file except in compliance with the License. A copy of the license can be found in the
installation directory, in the c:/OpenEdge/licenses folder. Software distributed under the
License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either
express or implied. See the License for the specific language governing rights and
limitations under the License. The Original Code is xs3p. The Initial Developer of the
Original Code is DSTC. Portions created by DSTC are Copyright © 2001, 2002 DSTC Pty
Ltd. All rights reserved.

OpenEdge includes YAJL software Copyright 2007, Lloyd Hilaiel. Redistribution and use
in source and binary forms, with or without modification, are permitted provided that the
following conditions are met: 1. Redistributions of source code must retain the above
copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in
binary form must reproduce the above copyright notice, this list of conditions and the
following disclaimer in the documentation and/or other materials provided with the
3. Neither the name of Lloyd Hilaiel nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE AUTHOR `AS IS' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

December 2009
Introducing OpenEdge Architect

OpenEdge® Architect 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

OpenEdge Architect 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 OpenEdge Architect. See the Eclipse Plugin Central Web site for more information.

Choose one of the following introductory topics for more information about OpenEdge Architect and its components:

- How to get started with OpenEdge Architect
- OpenEdge Projects
- AppServer
- ABL Editor
- Visual Designer
- Class Browser
- Running and Debugging ABL Programs
- AppBuilder and Other OpenEdge Tools
- Meta Catalog
- Tools for Business Logic
- Customization
What's New in Release 10.2B

The 10.2B release of OpenEdge Architect includes 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 Web site, 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 ref-
Getting Started

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.

**Related Reference**

- New and Noteworthy in Release 10.2A
- New and Noteworthy in Release 10.1C
Getting Started

New and Noteworthy in Release 10.2A

The 10.2A Release of OpenEdge Architect 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.

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

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

What's New in Release 10.2A

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.
- 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 sig-
nature 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 aaChar22 AS CHARACTER NO-UNDO.
DEFINE VARIABLE aaaChar333 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 aaChar22 AS CHARACTER NO-UNDO.
DEFINE VARIABLE aaaChar333 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.
- **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.

**Related Reference**

- What's New in Release 10.2B
- New and Noteworthy in Release 10.1C
New and Noteworthy in Release 10.1C

This section describes the new features and changes that were implemented in the 10.1C release of OpenEdge Architect.

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:

| ALT+SHIFT+T | Adds a static constructor to the current file. |
| ALT+SHIFT+Z, C | Surrounds the currently selected code with a CATCH block. |
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.

Related Reference

What's New in Release 10.2B
New and Noteworthy in Release 10.2A
Concepts

This section includes the following topics:

Eclipse terminology
Preferences and properties
OpenEdge Architect tools
Database connection profiles
The ABL Virtual Machine (AVM)
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 OpenEdge Architect, 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. OpenEdge Architect offers several predefined perspectives integrating diverse tools that combine to support a broader function, such as code editing, debugging, or database design.

Related Concepts

OpenEdge Architect tools

Related Tasks

Working with OpenEdge projects
Working with OpenEdge perspectives

Related Reference

OpenEdge perspectives and views
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>OpenEdge Architect>Shared AVM).

Related Concepts

Eclipse terminology
The ABL Virtual Machine (AVM)

Related Tasks

Setting workspace preferences
Setting project properties

Related Reference

OpenEdge Project properties
OpenEdge workspace preferences
Shared AVM preference page
OpenEdge Architect tools

OpenEdge Architect tools help you develop ABL applications. More specifically, they provide support for:

- Writing, reviewing, testing, and debugging code in the ABL language
- Searching a Meta Catalog of the entire code base, including the files of collaborators working on remote machines as well as your own local files, for specific content
- Viewing and modifying the structure of underlying databases
- Diagramming reusable Business Logic components, which are saved in an application model

A brief summary of each OpenEdge Architect tool follows. For detailed information, see the online help for the tool set that you are interested in.

**ABL Editor**

The central component in the OpenEdge Editor perspective is the ABL Editor, which offers code-entry shortcuts, syntax help, and color-coding and formatting options. Supporting use of the Editor, the Outline view aids analysis and navigation of the existing code, and a DB Structure view shows the schema of connected databases and offers drag-and-drop capability for entering table and field references.

**Visual Designer**

The Visual Designer is a graphical editor for building user interfaces for OpenEdge applications. These interfaces are written in ABL but are based on .NET controls, which are objects that contribute to a rich and robust GUI that has a contemporary look and feel. In Visual Designer, you drag and drop .Net controls onto a form on a design canvas.

A GUI built in ABL with .NET forms and controls is often referred to as the OpenEdge Advanced GUI.

**Debugger**

The OpenEdge Architect Debugger provides a full set of tools for testing and debugging ABL programs. The Debugger is an integral OpenEdge Architect component that works directly with the ABL Editor, making it easy to identify and fix problems as you work.

**Class Browser**

The Class Browser is an OpenEdge Architect 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).

**The AppBuilder and other tools**

The AppBuilder can run embedded in the OpenEdge Architect environment. The AppBuilder perspective provides graphical editing capabilities for UI development. Changes made in the AppBuilder are synchronized with the ABL Editor, and vice versa.
In addition, you can launch other OpenEdge tools (such as Data Dictionary, Data Administration, and PRO*Tools). These tools start in their own windows and are not embedded in the OpenEdge Architect environment.

**Progress DB Navigator**

The Progress DB Navigator allows you to view the schema of a database, browse the data in tables, view and analyze application data, and issue SQL commands. Progress DB Navigator connects to any SQL-compliant database through a JDBC driver.

**Meta Catalog**

The OpenEdge Meta Catalog allows you to build catalogs (databases) of information about the contents of an entire code base. By searching or browsing these catalogs, you can easily locate procedures, functions, references, and user-defined annotations.

You can define one or more local catalogs for projects in your workspace, which enable you to search the metadata for the resources under your control. You can also connect to a master catalog, which is defined by an administrator and synchronized with your group’s code management application. The master catalog indexes the code from all contributing developers and makes the information available to everyone working on the project.

The Meta Catalog is not associated with a specific perspective. Its search function is integrated with the standard Eclipse search mechanism. The Meta Catalog Explorer view, which you can show at any time, lets you browse all available catalogs.

**Tools for Business Logic**

The OpenEdge Tools for Business Logic perspective features a graphical editor called the Component Designer. With this tool, you create diagrams of application components by dragging elements onto the editing canvas from either a palette or from the Outline view of an ABL resource. You can automatically generate ABL code from a diagram, and conversely, you can create a diagram from existing code.

You can use the Component Designer to create diagrams of temp tables and ProDataSets, and to generate the corresponding ABL code.

Related components are grouped in an application model, stored as either an EMF Ecore file (see http://www.eclipse.org/emf/) or an OpenEdge database. The application model can be readily shared, allowing re-use of its components.

**Related Concepts**

  - Eclipse terminology

**Related Tasks**

  - Getting help

**Related Reference**

  - OpenEdge perspectives and views
Database connection profiles

**Demo**

*Defining Workspace Database Connections*

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 OpenEdge Architect.

**Related Tasks**

- Adding a new database connection to a project
- Adding existing database connections to a project
- Exporting and importing database connections

**Related Reference**

- Database preferences
- Add/Edit Connection Profile wizard
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 OpenEdge Architect, 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>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>OpenEdge>Shared OpenEdge 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...`).

**Related Concepts**
- Preferences and properties

**Related Tasks**
- Setting the default AVM startup parameters
- Setting up a shared AVM
Related Reference

- Shared AVM preference page
- OpenEdge settings
- Startup preference page
Tasks

This section includes the following topics:

- How to get started with OpenEdge Architect
- Starting OpenEdge Architect
- Setting workspace preferences
- Working with OpenEdge perspectives
- Getting help
How to get started with OpenEdge Architect

Getting started with OpenEdge Architect 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 OpenEdge Architect:

- Starting OpenEdge Architect
- Setting workspace preferences
- Working with OpenEdge perspectives
- Getting help

Once OpenEdge Architect is running, either open an existing project or create a new project.

Related Concepts

- Eclipse terminology
- Preferences and properties
- OpenEdge Architect tools
- Database connection profiles
- The ABL Virtual Machine (AVM)

Related Tasks

- Starting OpenEdge Architect
- Setting workspace preferences
- Working with OpenEdge perspectives
- Getting help
- Introducing OpenEdge Projects

Related Reference

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

**Demo**
Creating Your First OpenEdge Architect Workspace and Project

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

In Windows, you typically use the OpenEdge Architect icon in the Start menu, which is configured during installation to point to the correct Eclipse executable and JRE.

**Note:** One of the icons on the Start menu, OpenEdge Architect Clean, includes the -clean option in the startup command. The -clean option clears any cached data before starting the Eclipse executable. This is useful when restarting Eclipse after modifying the framework. It ensures that you see the latest version of all files.

In general, the OpenEdge Architect 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:
   a. If you specified a new workspace, Eclipse opens displaying the Welcome to OpenEdge Architect page.
   b. 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.

**Related Tasks**

Setting Eclipse startup preferences
Selecting a workspace
Setting workspace preferences
Setting Eclipse startup preferences

When you start OpenEdge Architect from a desktop icon or from a **Start** menu selection, the default Eclipse startup command has the following format:

```
OpenEdge-install-dir/oeide/eclipse/eclipse.exe -vm
OpenEdge_install_dir/jre/bin/javaw.exe
```

The default startup command points to the Eclipse executable (**eclipse.exe**) and the Java Runtime Environment (**javaw.exe**) that support OpenEdge Architect.

You might need to add startup parameters to the default so that OpenEdge Architect 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>
```

**Notes:** The **-vmargs** parameter passes any arguments that follow it directly to the Java Virtual Machine. Therefore, it must be placed at the end of the command line.

Setting the memory size to a value greater than the available physical memory causes thrashing.

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.

**Related Reference**

In Running Eclipse (in the Eclipse **Workbench User Guide**), scroll to *Advanced Topics in Running Eclipse* for a complete list of startup parameters.
Selecting a workspace

When you start OpenEdge Architect, 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 OpenEdge Architect is running by selecting File>Switch Workspace from the main menu bar. Eclipse restarts OpenEdge Architect when you change workspaces. If you want to work in multiple workspaces, you must start a separate instance of OpenEdge Architect for each workspace.

Related Concepts
- Eclipse terminology

Related Tasks
- Starting OpenEdge Architect

Related Reference
- Files (in the Eclipse Workbench User's Guide)
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.

Related Concepts

Preferences and properties
Eclipse terminology

Related Tasks

Setting the default AVM startup parameters
Setting up a shared AVM

Related Reference

OpenEdge workspace preferences
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 **OpenEdge** node.

3. Select the **Startup** node.

4. Specify the desired port ranges and startup parameters.

   **Note:** 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.

**Related Concepts**

The ABL Virtual Machine (AVM)

**Related Tasks**

Starting OpenEdge Architect

**Related Reference**

Startup preference page
Shared AVM preference page
Setting up a shared AVM

To configure a shared AVM:

1. Select **Window>Preferences** from the main menu bar of OpenEdge Architect.
2. Open **OpenEdge Architect>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.

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

**Related Concepts**

The ABL Virtual Machine (AVM)

**Related Tasks**

Starting OpenEdge Architect

**Related Reference**

Shared AVM preference page  
Startup preference page  
OpenEdge project properties
Working with OpenEdge perspectives

**Demos**
Using and Customizing Perspectives
Using and Managing Views

The perspectives included with OpenEdge Architect are AppBuilder, AppServer, Debugger, DB Navigator, Editor, Tools for Business Logic, and Visual Designer.

The topics in this section include:

- Opening a perspective
- Modifying a perspective
- Setting a perspective to its default state

**Related Concepts**
Eclipse terminology

**Related Reference**
OpenEdge perspectives and views
Opening a perspective

**Demos**

Using and Customizing Perspectives
Using and Managing Views

To open an OpenEdge perspective in the Eclipse Workbench:

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

**Related Concepts**

OpenEdge Architect tools
Eclipse terminology

**Related Tasks**

Modifying a perspective
Setting a perspective to its default state

**Related Reference**

OpenEdge perspectives and views
Modifying a perspective

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; OpenEdge Architect views are in folders whose names begin with "OpenEdge."

   **Note:** The views in the **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 OpenEdge Architect perspectives unchanged so that you retain the ability to return to a default state.

**Related Concepts**

- OpenEdge Architect tools
- About the Eclipse platform

**Related Tasks**

- Opening a perspective
- Setting a perspective to its default state

**Related Reference**

- OpenEdge perspectives and views
Setting a perspective to its default state

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.

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.

**Related Concepts**
- OpenEdge Architect tools
- About the Eclipse platform

**Related Tasks**
- Modifying a perspective
- Opening a perspective

**Related Reference**
- OpenEdge perspectives and views
Getting help

This section describes how to locate and use the help that is available for OpenEdge Architect, including:

The OpenEdge Architect Guide
Help display options
Searching
Context sensitive help
Context-sensitive help for .NET controls
Visual Designer tutorials
Cheat sheets
Demos
OpenEdge information resources
Eclipse information resources
The OpenEdge Architect Guide

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

The content of the *OpenEdge Architect Guide* is organized under the following topics:

- **Getting Started** - Contains help on basic topics, such as starting OpenEdge Architect, 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 OpenEdge Architect.
- **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 OpenEdge Architect 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.
- **AppBuilder and other OpenEdge tools** - 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 OpenEdge Architect 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 *OpenEdge Architect Guide*, the *Progress DB Navigator Guide* provides help on a tool you can use to manage, explore, and connect to databases.

**Related Topics**

- Help display options
- Searching
- Context-sensitive help
- Context-sensitive help for .NET controls
- Visual Designer tutorials
- Cheat sheets
- Demos
- OpenEdge information resources
- Eclipse information resources
Help display options

To change the way help is displayed, select **Windows>Preferences** from the main menu bar of OpenEdge Architect. 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*.

**Related Topics**

The OpenEdge Architect Guide
Searching
Context-sensitive help
Context-sensitive help for .NET controls
Visual Designer tutorials
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.

**Related Topics**
- The OpenEdge Architect Guide
- Help display options
- Context-sensitive help
- Context-sensitive help for .NET controls
- Visual Designer tutorials
- Cheat sheets
- Demos
- OpenEdge information resources
- Eclipse information resources
Getting Started

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 OpenEdge Architect 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 OpenEdge Architect topics appearing in the dynamic help list.

**Note:** Context-sensitive help for .NET controls in the Visual Designer is available when you install the Infragistics NetAdvantage .NET Help from OpenEdge Documentation and Samples.

**Related Topics**

- The OpenEdge Architect Guide
- Help display options
- Searching
- Context-sensitive help for .NET controls
- Visual Designer tutorials
- Cheat sheets
- Demos
- OpenEdge information resources
- Eclipse information resources
Context-sensitive help for .NET controls

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.

**Note:** The help for .NET controls appears in the Microsoft Document Explorer instead of the browser that displays OpenEdge Architect Eclipse help. The Microsoft Document Explorer is automatically installed when you install OpenEdge Architect.

**Related Topics**

- Getting control-specific information
- The OpenEdge Architect Guide
- Help display options
- Searching
- Context-sensitive help
- Visual Designer tutorials
- Cheat sheets
- OpenEdge information resources
- Eclipse information resources
Visual Designer tutorials

You can find tutorials that explain the features and functions of the OpenEdge Architect Visual Designer in *OpenEdge Getting Started: Introducing the OpenEdge Architect Visual Designer*.

**Related Topics**

- The OpenEdge Architect Guide
- Help display options
- Searching
- Context-sensitive help
- Context-sensitive help for .NET controls
- Cheat sheets
- Demos
- OpenEdge information resources
- Eclipse information resources
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 OpenEdge Architect. Choose Help>Cheat Sheets from the main menu bar to view the list of available cheat sheets.

Related Topics

- The OpenEdge Architect Guide
- Help display options
- Searching
- Context-sensitive help
- Context-sensitive help for .NET controls
- Visual Designer tutorials
- Demos
- OpenEdge information resources
- Eclipse information resources
Demos

There are a variety of demos, hosted on the PSDN 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 at OpenEdge Architect Videos: Using Architect Collection.

For general information about both OpenEdge and OpenEdge Architect, see the OpenEdge Tour.

Related Topics
- The OpenEdge Architect Guide
- Help display options
- Searching
- Context-sensitive help
- Context-sensitive help for .NET controls
- Visual Designer tutorials
- Cheat sheets
- OpenEdge information resources
- Eclipse information resources
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

**OpenEdge Architect 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**

**Related Topics**
- The OpenEdge Architect Guide
- Help display options
- Searching
- Context-sensitive help
- Context-sensitive help for .NET controls
- Visual Designer tutorials
- Cheat sheets
- Demos
- 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 OpenEdge Architect (Help>Help Contents). For newsgroups, documentation, and articles about Eclipse, go to the eclipse.org Web site.

Related Topics

The OpenEdge Architect Guide
Help display options
Searching
Context-sensitive help
Context-sensitive help for .NET controls
Visual Designer tutorials
Cheat sheets
Demos
OpenEdge information resources
Reference

This section includes the following topics:

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

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 Preference 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).

**Related Concepts**
- Eclipse terminology
- Preferences and properties

**Related Tasks**
- Setting workspace preferences

**Related Reference**
- Classic Software Updates
- Views preference page
- Database Connections preference page
- Shared AVM preference page
- Preference pages for OpenEdge tools
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 **OpenEdge Architect** and select **Views** to open the Views preference page.

The following options are available on the Views preference page:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Show compile log in Console view</strong></td>
<td>Display build messages in the standard Eclipse Console view. No build messages are displayed when this check box is cleared.</td>
</tr>
<tr>
<td><strong>Show informational messages in Console view</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Show warning messages in Console view</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Split functions and procedures in Outline view</strong></td>
<td>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 <strong>Code Sections</strong>. 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).</td>
</tr>
<tr>
<td><strong>Show procedure libraries in Resources view</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Show warning message when running procedures that contain errors</strong></td>
<td>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 <strong>Do not show this warning again</strong> option. (You can re-enable warnings after suppressing them by selecting this option again.)</td>
</tr>
</tbody>
</table>

**Related Concepts**

- Eclipse terminology
- Database connection profiles
**Related Tasks**
- Working with perspectives

**Related Reference**
- OpenEdge perspectives and views
- OpenEdge workspace preferences
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 **OpenEdge Architect** and select **Startup** to open the **Startup** preference page.

The following options are available on the Startup preference page:

<table>
<thead>
<tr>
<th><strong>Lower Port Number, Upper Port Number</strong></th>
<th>The range of ports on which OpenEdge Architect 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.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Default startup parameters</strong></td>
<td>Any appropriate parameters for running the OpenEdge AVM. Refer to <em>OpenEdge Deployment: Startup Command and Parameter Reference</em> for detailed information on startup parameters. You can find the manual in the <strong>Product Documentation</strong> 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 (<strong>Project&gt;Properties&gt;OpenEdge</strong>). You can also override or append to these parameters on the Shared AVM preferences page (<strong>Window&gt;Preferences&gt;Shared OpenEdge AVM</strong>). <strong>Note:</strong> Do not specify database connections as startup parameters. Use the <strong>Database Connections</strong> page for this purpose. You can use a parameter file by including <code>-pf parameter_file_path</code>. 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.</td>
</tr>
</tbody>
</table>

**Related Tasks**

- Setting workspace preferences
- Setting project properties

**Related Reference**

- OpenEdge workspace preferences
- Database Connections preference page
Database Connections preference page

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:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>New</strong></td>
<td>Add a new connection profile where you can define a new workspace database connection. For more information, see Add/Edit Connection Profile wizard.</td>
</tr>
<tr>
<td><strong>Edit</strong></td>
<td>Modify 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><strong>Copy</strong></td>
<td>Copy an existing connection profile.</td>
</tr>
<tr>
<td><strong>Remove</strong></td>
<td>Deletes a connection profile.</td>
</tr>
<tr>
<td><strong>Import</strong></td>
<td>Import 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 Exporting and importing database connections.</td>
</tr>
<tr>
<td><strong>Export</strong></td>
<td>Export 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 and importing database connections.</td>
</tr>
</tbody>
</table>

**Related Tasks**

- Adding existing database connections to a project
- Adding a new database connection to a project

**Related Reference**

- OpenEdge database connection wizard
**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 **OpenEdge Architect** 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><strong>Working directory</strong></th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Temp directory</strong></td>
<td>The directory in which temporary files are created. By default, this is the working directory specified when OpenEdge Architect is installed.</td>
</tr>
</tbody>
</table>
| **Startup parameters**| 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 OpenEdge Architect 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 OpenEdge Architect Visual Designer preference page. |
**Getting Started**

| 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 OE Import or OE Export wizard, expand the OpenEdge node and select Shared AVM Settings. The wizard allows you to specify a source or destination for the settings.

**Related Concepts**

- Preferences and properties
- The ABL Virtual Machine (AVM)
- Database connection profiles

**Related Tasks**

- Setting workspace preferences

**Related Reference**

- Shared AVM properties
- Projects page
- OpenEdge workspace preferences
- OpenEdge project 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.

**Related Reference**

- AppBuilder properties page
- Assemblies properties page
- Database connections properties page
- Projects page
- PROPATH and Source properties page
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.

**Related Concepts**
- The ABL Virtual Machine (AVM)

**Related Tasks**
- Setting workspace preferences

**Related Reference**
- OpenEdge workspace preferences
- Shared AVM preference page
Preference pages for OpenEdge tools

The following topics contain information on the preference pages for various OpenEdge tools:

- Class Cache
- Customization
- Launching
- Debug
- Editor
- Annotations
- Assistance
- Build
- Colors
- Templates (Macros)
- Meta Catalog
- Server
- Tools for Business Logic
- Visual Designer

**Concepts**

- Eclipse terminology
- OpenEdge workspace preferences

**Tasks**

- Setting workspace preferences

**Reference**

- OpenEdge workspace preferences
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** preference dialog.

This section describes the following pages in the OpenEdge database connection wizard:

- **Add/Edit Connection Profile**
- **Define a SQL connection**
- **Add new SQL connection**
- **Define database server configuration**
Add/Edit OpenEdge database connection profile wizard

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>
<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 OpenEdge Architect). You cannot connect to an OpenEdge Personal RDBMS database that is running remotely.</td>
</tr>
<tr>
<td></td>
<td>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 <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>User ID</strong></td>
<td>If the user ID and the password were previously created, enter the information in the <strong>User ID</strong> field. You must specify a user ID.</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>The password to access the database.</td>
</tr>
<tr>
<td><strong>Aliases</strong></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>
</tbody>
</table>
### Group

Used for organizing your connection profiles. You can add a new group or you can select an existing group.

### Other Parameters

Displays a string of supported database connection parameters.

## Related Concepts

- Database connection profiles
Define a SQL connection profile

Demo
Defining Workspace Database Connections

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 <strong>Add new SQL connection</strong> radio button indicates that the connection profile uses a new SQL connection. Select this option, then click <strong>Next</strong> to display the <strong>Add New SQL Connection Profile</strong> page.</td>
</tr>
<tr>
<td>Use existing SQL connection</td>
<td>The <strong>Use existing SQL connection</strong> 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 <strong>Next</strong>. The next page in the connection profile wizard (<strong>Define Database Server configuration</strong>) appears. <strong>Note:</strong> When you select the <strong>Use existing SQL connection</strong> radio button items in the browse become selectable.</td>
</tr>
</tbody>
</table>

**Related Concepts**

- Database connection profiles

**Related Reference**

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

**Demo**

**Defining Workspace Database Connections**

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><strong>Connection name</strong></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><strong>Driver</strong></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 Configure a driver.</td>
</tr>
<tr>
<td><strong>User ID</strong></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><strong>Password</strong></td>
<td>Specifies the password to access the database.</td>
</tr>
<tr>
<td><strong>Open on startup of OpenEdge Architect</strong></td>
<td>Opens the database connection when OpenEdge Architect starts.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> If you enable this option, you still must enter a valid user ID and an optional password. This option slows the start-up of Eclipse. If you select the <strong>Auto-start a database server</strong> on the next page of database connection wizard, you can not use this option; the system will attempt to connect before the database automatically starts.</td>
</tr>
<tr>
<td><strong>Auto-Login on connect</strong></td>
<td>Automatically logs into the database when OpenEdge Architect starts. The system attempts to log in using the provided user ID and password.</td>
</tr>
<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>
</tbody>
</table>
The host name must be `localhost` when you are using the OpenEdge Personal RDBMS database (which is the default RDBMS packaged with OpenEdge Architect). 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 `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.

**Database name**

Represents the name of the database.

*Note:* If you are connecting to an OpenEdge database, check for the correct name in the **Databases** folder in Progress Explorer.

**SQL parameters**

Displays any additional SQL parameters delimited by a semi-colon in the format `key1=value; key2=value`.

---

**Related Concepts**

Database connection profiles

---

**Related Tasks**

Adding existing database connections to a project
Adding a new database connection to a project

---

**Related Reference**

Database connections preferences

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.
Define an OpenEdge database server configuration

This page appears when you define the database server configuration when OpenEdge Architect 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><strong>Auto-start database server</strong></td>
<td>Automatically starts the database server when you start OpenEdge Architect. Use this option only if you are starting a local database.</td>
</tr>
<tr>
<td><strong>Physical name</strong></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><strong>Service/Port</strong></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><strong>Parameters</strong></td>
<td>Specifies a string of supported database connection parameters.</td>
</tr>
<tr>
<td><strong>Auto-shutdown database server</strong></td>
<td>Automatically stops the database server when you quit OpenEdge Architect.</td>
</tr>
</tbody>
</table>
OpenEdge perspectives and views

Demos
Using and Customizing Perspectives
Using and Managing Views

The following topics summarize the perspectives and views that are available with OpenEdge Architect:

- OpenEdge default perspectives - A list of views included by default in each perspective.
- OpenEdge views - A brief description of each view

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

Related Concepts
OpenEdge Architect tools
The Eclipse platform

Related Tasks
Opening a perspective
Modifying a perspective
Setting a perspective to its default state
OpenEdge default perspectives

OpenEdge perspectives are available when you select **File>**New>Open Perspective** from the main menu bar of OpenEdge Architect.

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 OpenEdge Architect Guide</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</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>AppBuilder and Other OpenEdge Tools</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>OpenEdge AppServer</td>
<td>Resources</td>
<td>AppServer</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>Properties</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></td>
<td>Servers</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OpenEdge Server Monitor</td>
<td></td>
</tr>
</tbody>
</table>
### Related Concepts

- OpenEdge Architect tools
- The Eclipse platform

### Related Tasks

- Opening a perspective
- Modifying a perspective
- Setting a perspective to its default state

### Related Reference

- OpenEdge views
OpenEdge views

**Demo**

**Using and Managing Views**

Views are organized in categories corresponding to the folders in the Show View window that appears when you select Window>Show View>Other. Within each category, views are listed in alphabetical order.

The following table provides a brief description of the OpenEdge views and the standard Eclipse views that appear by default in one or more OpenEdge perspectives.

<table>
<thead>
<tr>
<th>Category</th>
<th>View</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Progress DB Navigator</td>
<td>Connection Info</td>
<td>Shows details pertaining to the currently active session selected in the Connections view.</td>
</tr>
<tr>
<td></td>
<td>Connections</td>
<td>Lists currently active connections and available connection profiles; lets you create databases, define connection profiles, establish connections, and launch SQL editors.</td>
</tr>
<tr>
<td></td>
<td>DB Details</td>
<td>Provides detailed information about the schema element selected in the DB Structure view.</td>
</tr>
<tr>
<td></td>
<td>DB Structure</td>
<td>Shows a structural outline of the schema of each connected database.</td>
</tr>
<tr>
<td></td>
<td>Navigator</td>
<td>Lets you select projects and resources to operate on; shows the file system structure and contents of the project root directory and its subdirectories.</td>
</tr>
<tr>
<td></td>
<td>SQL History</td>
<td>Maintains a record of SQL statements executed in the current session.</td>
</tr>
<tr>
<td></td>
<td>SQL Results</td>
<td>Displays the results of SQL statements.</td>
</tr>
<tr>
<td>OpenEdge Debugger</td>
<td>Breakpoints</td>
<td>Lets you add, edit, remove, enable, and disable breakpoints.</td>
</tr>
<tr>
<td></td>
<td>Dynamic Objects</td>
<td>Lets you monitor the creation and the destruction of dynamic object instances in the ABL application.</td>
</tr>
<tr>
<td></td>
<td>Debug</td>
<td>Displays a session tree, including the ABL stack trace for each active debugging session.</td>
</tr>
<tr>
<td></td>
<td>Variables</td>
<td>Shows the values of variables, parameters, buffers, temp tables, and datasets associated with the current procedure.</td>
</tr>
<tr>
<td></td>
<td>Expressions</td>
<td>Displays values of user-specified variables, buffer fields, attributes, data, and built-in ABL functions.</td>
</tr>
</tbody>
</table>
### OpenEdge Editing

<table>
<thead>
<tr>
<th>OpenEdge Editing</th>
<th>Outline</th>
<th>Displays a tree view of the components of the currently active file, including variables, functions, and procedures.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DB Structure</td>
<td>Shows the structure of all databases connected to project containing the currently selected resource.</td>
</tr>
<tr>
<td></td>
<td>Problems</td>
<td>Displays errors and warnings.</td>
</tr>
<tr>
<td></td>
<td>Resources</td>
<td>Lets you select projects and resources to operate on; shows the file system structure and contents of the project root directory and its subdirectories.</td>
</tr>
</tbody>
</table>

### OpenEdge Meta Catalog

| OpenEdge Meta Catalog | Meta Catalog Explorer | Lets you browse the contents of all available catalogs. |

### OpenEdge Support Views **Note:** Do not select these views from the Show View list. They appear automatically in conjunction with specific operations.

<table>
<thead>
<tr>
<th>OpenEdge Support Views</th>
<th>ABL Display</th>
<th>Displays results when ABL procedures are executed.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AppBuilder Framework</td>
<td>Displays AppBuilder tools.</td>
</tr>
<tr>
<td></td>
<td>Keyword Help</td>
<td>Displays help when you press Shift+F2 or when you right-click and select <strong>Keyword Help</strong>.</td>
</tr>
<tr>
<td></td>
<td>WebSpeed</td>
<td>Displays results when you execute a WebSpeed application.</td>
</tr>
</tbody>
</table>

### Basic (Standard Eclipse views)

<table>
<thead>
<tr>
<th>Basic (Standard Eclipse views)</th>
<th>Console</th>
<th>Displays various system messages.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Outline</td>
<td>Displays logical elements of the file currently being edited.</td>
</tr>
<tr>
<td></td>
<td>Problems</td>
<td>Displays errors and warnings.</td>
</tr>
<tr>
<td></td>
<td>Properties</td>
<td>Provides details about various selected items (for example, application model elements).</td>
</tr>
<tr>
<td></td>
<td>Tasks</td>
<td>Shows a user-defined to-do list.</td>
</tr>
</tbody>
</table>

### Related Concepts

- OpenEdge Architect tools
- The Eclipse platform

### Related Tasks

- Opening a perspective
- Modifying a perspective
- Setting a perspective to its default state

### Related Reference

- OpenEdge default perspectives
Classic Software Updates

Eclipse Release 3.4 introduced a completely new software update feature. The new feature determines all the software prerequisites and installs them automatically. The new software update feature starts when you choose Help>Software Updates from the main menu bar of OpenEdge Architect.

The check box on the Classic Software Updates preference page allows you to enable the software update feature that existed in previous versions of Eclipse. When the feature is enabled, you can choose Help>Classic Software Updates from the main menu bar of OpenEdge Architect.

**Note:** Enabling Classic Software Updates does not disable the new software update feature. You will be able to choose either the new or the classic update feature from the Help menu.

**Related Reference**

New Features in Eclipse 3.4
OpenEdge workspace preferences
Introducing OpenEdge Projects

OpenEdge Architect, 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.

OpenEdge Projects in OpenEdge Architect have properties that specifically support ABL development. OpenEdge Projects include properties such as build and PROPATH settings, and database connections.
Concepts

This section includes the following topics:

- OpenEdge Architect project and resource data files
- Configuration variables

OpenEdge Architect project and resource data files

When you are working in an OpenEdge project, OpenEdge Architect 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 Resource 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.

**Caution:** 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 OpenEdge Architect 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>
<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>

**Related Concepts**
- Code annotations
- Database connection profiles
- .NET assemblies and the Visual Designer

**Related Tasks**
- Adding database connections to a project
- Adding Sonic ESB annotations
- Adding and removing assembly references
- Deleting a container
Sharing Toolbox settings

Related Reference

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 variables are valid:

- `{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 OpenEdge page. (You can enter a period (.) to represent `{WORK}`.)
- `{RCODE}` - The directory (if any) specified in the Build destination field on the Build 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 on the Build properties page.
- In the alias directory entries on the WebSpeed properties page.
- In Propath and Source entries in the Propath properties page.
- In entries on the References dialog in the ABL Code Editor. (Use CTRL+SHIFT+G in the editor to see the References dialog.)

**Related Tasks**

- Creating a new project
- Reusing project properties
- Sharing project settings

**Related Reference**

- OpenEdge settings
- AppBuilder settings
- Build settings
- Custom settings
- Database Connections settings
- PROPATH settings
- WebSpeed settings
Tasks

This section includes the following topics:

- Creating a new project
- Adding resources to a project
- Adding existing database connections to a project
- Adding a new database connection to a project
- Setting project properties
- Using project property profiles
- Exporting and importing resources and settings
Creating a new project

To create an OpenEdge project:

1. Select New>OpenEdge Project from the File menu or the Resources view context (right-click) menu.

   Be sure to create an OpenEdge Project. Do not create a General Project or any other available type. OpenEdge projects are customized to take full advantage of OpenEdge Architect features.

2. Type a name for the new project.

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

3. If you want to accept all the default properties for the project, click Finish now.

   Otherwise, see New OpenEdge Project wizard for a description of the options for changing the default project properties.

4. To review or modify project properties after the project is created, right-click the project in the Resources view and select Properties.

Note: For large applications with multiple modules, it is usually best to create separate projects. You must have separate projects for modules that have a different PROPATH or that require different database connections.

Related Tasks

Adding resources to a project
Adding existing database connections to a project
Adding a new database connection to a project
Setting project properties
Importing project settings and resources

Related Reference

Project property settings
New OpenEdge Project wizard
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 **Resources** 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:
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.

**Related Concepts**

Eclipse terminology

**Related Tasks**

Creating a new project
Adding existing database connections to a project
Adding a new database connection to a project
Setting project properties
Exporting and importing resources and settings
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 Resources 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 OpenEdge Architect 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 checkboxes next to the databases that you want to add to the project.
8. Click OK.

Related Task
- Creating a new project
- Adding new resources to a project
- Adding a new database connection to a project
- Setting project properties
- Importing project settings and resources

Related Reference
- Database Connections settings
Adding a new database connection to a project

To create a new database connection:

1. From the Resources 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 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.

Related Tasks

Creating a new project
Adding new resources to a project
Adding existing database connections to a project
Setting project properties
Importing project settings and resources

Related Reference

Database Connections properties page
Add Connection Profile wizard
Setting project properties

To review the project properties and modify them if necessary:

1. Right-click the project in the Resources view and select Properties from the context menu.
2. In the left pane of the Properties page, expand 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 OpenEdge page and selecting the profile.

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

**Related Tasks**
- Creating a new project
- Adding new resources to a project
- Adding existing database connections to a project
- Adding a new database connection to a project
- Importing project settings and resources

**Related Reference**
- OpenEdge project properties pages
- Configuration variables
- OpenEdge settings
- AppBuilder settings
- Build settings
- Modules properties page
- Custom settings
- Database Connections settings
- Propath settings
- WebSpeed settings
Using 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 Resources view and select Properties from the context menu.
2. In the left pane of the Properties window, select OpenEdge.
3. Select the Custom page and click Save Profile.
4. Enter a name for the profile and click OK.

To change to a different project property profile:

1. Right-click the project folder in the Resources view and select Properties from the context menu.
2. In the left pane of the Properties window, select 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.

**Related Tasks**
- Setting project properties
- Sharing project settings

**Related Reference**
- Custom settings
- OpenEdge settings
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
Importing files and folders
Exporting and importing projects
Exporting and importing database connections
Importing breakpoints
Migrating database connections
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 Resources view and select Properties from the context menu.
2. In the left pane of the Properties window, select 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.

**Related Tasks**

- Setting project properties
- Saving a named project profile

**Related Reference**

- Custom settings
Importing resources

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

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

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

- 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.

**Related Concepts**

Resources (in the Eclipse Workbench User Guide)

**Related Tasks**

Adding new resources to a project

**Related Reference**

New Folder wizard (in the Eclipse Workbench User Guide)
Exporting and 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.

However, for OpenEdge projects, you must do the following before exporting a project:

1. Click **Share Settings** on the Custom properties sheet to create a `project.xml` file in your project folder.
2. 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.

To create an archive of the project:

1. Select **File>Export>General>Archival File**.
2. 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.
3. Specify the path and the name of the archive file.
4. Select the archive format (tar or zip) and any other options you require.
5. Click **Finish**.

To import an archived project:

1. Start OpenEdge Architect 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.

**Related Tasks**

- Sharing project properties
- Exporting and importing database connections
- Importing breakpoints
Exporting and 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 export connection information:

1. Start OpenEdge Architect in the workspace that is the source of the connection information.
2. Select File>Export. The Export dialog appears.

   Note: You can also start the Export dialog from the Database Connections preference page.

3. Choose the 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.

To import connection information:

1. Start OpenEdge Architect in the workspace where the connection information will be imported.
2. Select File>Import. The Import dialog appears.

   Note: You can also start the Import dialog from the Database Connections preference page.

3. Choose the 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.

**Related Reference**

- Database Connections preferences
- Database Connections settings
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 OpenEdge Architect 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 OpenEdge Architect in the workspace where you want to import the breakpoint information.
9. Click Next and the Import Breakpoints dialog appears.
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.

Related Reference

Import wizard (in the Eclipse Workbench User Guide)
Export wizard (in the Eclipse Workbench User Guide)
Migrating database connections

Database connections created in Release 10.1A cannot be used in later releases of OpenEdge Architect. If you want to use a database connection created in Release 10.1A in this release of OpenEdge Architect, 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, OpenEdge Architect 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. In the Migrate Database Connections dialog, 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 an unique name for each instance. Select multiple instances when the connection definitions differ.

6. Click Finish.
Reference

This section includes the following topics:

- New OpenEdge Project wizard
- OpenEdge project properties pages
New OpenEdge Project wizard

The New OpenEdge Project wizard allows you to create Eclipse projects that specifically support ABL application development. ABL application development support includes project settings for the AVM, the AppServer, PROPATH, and database connections.

You can start the New OpenEdge Project wizard by selecting **File>New>OpenEdge Project** from the main menu bar.

The New OpenEdge Project wizard contains the following pages:

- Create an OpenEdge Project page
- Define AppServer module page
- Define PROPATH page
- Select database connections page

**Related Concepts**
- Database connection profiles
- The ABL Virtual Machine (AVM)

**Related Tasks**
- Creating a new project
Create an OpenEdge Project page

The Create an OpenEdge Project page of the New OpenEdge Project wizard allows you to set a number of general options for a new project. It contains the following controls:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project name</strong></td>
<td>Assign a name that is unique in the workspace to a folder that will contain project resources. <strong>Note:</strong> Avoid changing the project name outside of OpenEdge Architect (with Windows Explorer, for example). The project will become corrupted and you will be unable to reopen it. To change the name after the project is created, choose the project name in the Resources view and select <strong>File&gt;Rename</strong> from the main menu bar.</td>
</tr>
<tr>
<td><strong>Use default location</strong></td>
<td>Specify that the project folder will be created in the default location, which is the top level of the workspace folder.</td>
</tr>
<tr>
<td><strong>Location</strong></td>
<td>When <strong>Use default location</strong> is not selected, specifies the directory that will contain the project folder.</td>
</tr>
<tr>
<td><strong>Use project-specific AVM</strong></td>
<td>Specify that an AVM, dedicated solely to the project you are creating, is started whenever you open the current workspace. To change this setting after the project is created, use the OpenEdge properties page.</td>
</tr>
<tr>
<td><strong>Use shared AVM</strong></td>
<td>Specify use of the AVM that is available to all projects in the workspace. To change this setting after the project is created, use the OpenEdge properties page.</td>
</tr>
<tr>
<td><strong>Use TTY for runtime</strong></td>
<td>An option to run ABL procedures in a character client window rather than in a GUI window. The character client window opens when you open the workspace that contains the project. To change this setting after the project is created, use the OpenEdge properties page.</td>
</tr>
<tr>
<td><strong>Use project root directory for source and r-code</strong></td>
<td>Specify that source and r-code will be in the top-level of the project folder.</td>
</tr>
</tbody>
</table>
### Use separate source and r-code directories

Specify where 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 the specified source location and the specified r-code destination. If r-code is generated from source 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.

### Add AppServer module

Create a directory for the code that will be published to an AppServer. By default, the AppServer module is a folder named `AppServerContent`, which is created in the project folder. You can modify the default on the Modules property page when you finish creating the project. Also be aware that AppServer support in an OpenEdge project includes the creation of a `.settings` folder, which contains configuration information.

---

### Related Concepts

- OpenEdge Architect project and resource data files
- The ABL Virtual Machine (AVM)

### Related Tasks

- Creating a new project
- AppServer project support

### Related Reference

- Define AppServer module page
- Define PROPATH page
- Select database connections page
- OpenEdge project properties pages
Define AppServer module page

The Define AppServer module page of the New OpenEdge Project wizard appears only if you selected Add AppServer module from the Create an OpenEdge Project page. It allows you to define the project module and choose publishing options. It contains the following controls:

<table>
<thead>
<tr>
<th>Module name</th>
<th>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>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>Select where to publish from a list of available servers.</td>
</tr>
<tr>
<td>Publish changes immediately</td>
<td>Choose if you want to copy files to the server immediately after the project is created.</td>
</tr>
</tbody>
</table>

**Related Concepts**

- Project support for AppServers
- AppServer terminology

**Related Tasks**

- Creating a new project
- AppServer project support

**Related Reference**

- AppServer-related views, pages, and wizards
Define PROPATH page

The **Define PROPATH** page of the New OpenEdge Project wizard allows you to view and to modify the project’s PROPATH settings. After the project is created, you can change PROPATH settings from the PROPATH and Source property page. However, if the project uses the shared AVM, you must use the Shared AVM/PROPATH preference page.

**Related Reference**

- PROPATH and Source properties page
- Shared AVM preference pages


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 OpenEdge Architect 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.

Related Tasks

Adding a new database connection to a project

Related Reference

Database connections properties page
Shared AVM preference pages
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 Resources view and select Properties from the context menu, the Project menu, or the File menu.

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 OpenEdge Architect. The OpenEdge node contains project properties pages that apply specifically to OpenEdge projects, including:

- **OpenEdge properties page** - Allows you to view and adjust certain settings for the OpenEdge AVM.

- **AppBuilder properties page** - Allows you to choose whether the AppBuilder runs in an embedded Eclipse view (the AppBuilder Designer view) or in a detached window. It also allows you to enable the embedded AppBuilder for Progress Dynamics application development.

- **Assemblies properties page** - Allows you to specify the location and to modify the content of the assemblies.xml file.

- **Build properties page** - Allows you to view and to adjust options related to compiling ABL source code and saving r-code.

- **Custom 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.

- **Database connections properties page** - Allows you to view and to select the database connections for a project.

- **Modules properties page** - Allows you to view and to configure the module that contains the files that will be published to the AppServer.

- **PROPATH and Source properties page** - Allows you to control where the AVM searches for files and procedures, to specify the folders where source code resides, and to specify the build destination for r-code.

- **WebSpeed properties page** - Allows you to define the broker URL and the path settings for use with WebSpeed applications.

**Related Concepts**

Configuration variables

**Related Tasks**

Creating a new project
Setting project properties
The OpenEdge 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 OpenEdge page, right-click an OpenEdge project name in the Resources view and choose Properties from the context menu. Click the 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. <strong>Note:</strong> 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 OpenEdge Architect&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 OpenEdge Architect (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 OpenEdge Architect was installed.</td>
</tr>
</tbody>
</table>
### Startup parameters

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.

### 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 *OpenEdge Architect/Startup* preference page to change the defaults.

### Use TTY for runtime

An option to run ABL procedures in a character client window rather than in a GUI window. The character client window opens when you open the workspace that contains the project.

### 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.

---

**Related Tasks**

- Setting project properties

**Related Reference**

- Startup preference page
- Configuration variables
- AppBuilder settings
- Build settings
- Custom settings
- Database Connections settings
- PROPATH settings
- WebSpeed settings
AppBuilder properties page

The AppBuilder properties page allows you to choose whether the AppBuilder runs in an embedded Eclipse view (the AppBuilder Designer view) or in a detached window. It also allows you to enable the embedded AppBuilder for Progress Dynamics application development.

To access the AppBuilder page, right-click an OpenEdge project name in the Resources view and choose Properties from the context menu. Expand the 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 appear dimmed. They cannot be changed on this page, and they may also be inaccurate. You must go to the AppBuilder 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 OpenEdge Architect>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></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Start AppBuilder when switching to AppBuilder perspective</strong></td>
<td>Immediately launch AppBuilder when the AppBuilder perspective opens. By default, AppBuilder does not start until you either click on the AppBuilder button on the toolbar, select it from the OpenEdge Tools menu, or double click on a .w file in the Resources view.</td>
</tr>
<tr>
<td><strong>Start AppBuilder only if .w files are open</strong></td>
<td>Automatically start AppBuilder only when .w files are open in the ABL editor. This option can save time (especially during OpenEdge Architect startup) since it prevents unnecessary launching of AppBuilder.</td>
</tr>
<tr>
<td><strong>Enable Progress Dynamics</strong></td>
<td>Enable the AppBuilder for Progress Dynamics application development.</td>
</tr>
<tr>
<td><strong>Configuration File</strong></td>
<td>Specify the path of the XML configuration file where Dynamics session types are defined.</td>
</tr>
<tr>
<td><strong>Session Type</strong></td>
<td>Specify the name of the session type.</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.
**Related Tasks**

- Setting project properties
- Starting the AppBuilder
- Starting the AppBuilder with Dynamics enabled

**Related Reference**

- Configuration variables
- OpenEdge settings
- Build settings
- Custom settings
- Database Connections settings
- PROPATH settings
- WebSpeed settings
Assemblies properties page

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 OpenEdge Architect 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 Resources view and choose Properties from the context menu. Expand the 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 OpenEdge Architect>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><strong>Use default location</strong></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><strong>Workspace/File System</strong></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><strong>assemblies.xml file viewer</strong></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><strong>Add</strong></td>
<td>Open the Add Assembly References dialog, where you can add both global and local assemblies to the assemblies.xml file.</td>
</tr>
<tr>
<td><strong>Move Up/Down</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Remove</strong></td>
<td>Remove the assembly selected in the viewer from the <code>assemblies.xml</code> file.</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>When you delete a control with the Visual Designer, the assembly reference is not automatically removed from the <code>assemblies.xml</code> file. Although unused assembly references do no harm, you can delete them with this control.</td>
</tr>
</tbody>
</table>

**Related Concepts**

.NET assemblies and the Visual Designer  
The ABL Virtual Machine (AVM)

**Related Tasks**

- Adding and removing assembly references  
- Using the Class Browser

**Related Reference**

- Configuration variables  
- OpenEdge settings  
- Build settings  
- Custom settings  
- Database Connections settings  
- PROPATH settings  
- WebSpeed settings
Build properties page

The Build properties page allows you to view and adjust options related to compiling ABL source code and saving r-code.

To access the Build page, right-click an OpenEdge project name in the Resources view and choose Properties from the context menu. Expand the OpenEdge node in the tree view, and then click the Build node.

The following controls are available:

| **Save r-code** | Save .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. |
| **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. |
| **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. |
| **XCODE key** | The key for encrypted source files. Leave this field blank if you do not use encrypted source code. |
| **Pre-compile callback routine** | Specifies a procedure that runs prior to compilation. |
**Related Tasks**

Setting project properties  
Adding Sonic ESB annotations

**Related Reference**

OpenEdge properties page  
Database connections properties page  
Modules properties page  
PROPATH and Source properties page  
WebSpeed properties page  
Configuration variables

<table>
<thead>
<tr>
<th>Create cross-reference files in XML format</th>
<th>Save cross references in an XML file. This corresponds to the COMPILE option, XREF-XML. 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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>XREF XML destination</td>
<td>The path of the folder where cross references are saved.</td>
</tr>
<tr>
<td>Generate Sonic invocation files on build</td>
<td>Automatically generate Sonic .esboe files whenever a build occurs for the project.</td>
</tr>
<tr>
<td>Invocation file destination</td>
<td>The path of the folder where Sonic .esboe files will be saved.</td>
</tr>
</tbody>
</table>
Custom properties page

The Custom 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 Custom properties page, right-click an OpenEdge project name in the Resources view and choose Properties from the context menu. Expand the OpenEdge node in the tree view, and then click the Custom node.

The following controls are available:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@@{APP}</td>
<td>An optional variable that represents the root directory for your application, and is used in path statements. Enter the appropriate path.</td>
</tr>
<tr>
<td>@@{DB}</td>
<td>An optional variable that represents the root directory for your database, and is used in path statements. Enter the appropriate path.</td>
</tr>
<tr>
<td>Share Settings</td>
<td>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.</td>
</tr>
<tr>
<td>Import Settings</td>
<td>Imports a set of project properties from a previously generated properties file and applies them to the currently selected project.</td>
</tr>
<tr>
<td>Save Profile</td>
<td>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 OpenEdge properties page.</td>
</tr>
</tbody>
</table>

**Related Tasks**

- Setting project properties
- Importing project settings and resources

**Related Reference**

- Configuration variables
- OpenEdge settings
- AppBuilder settings
- Build settings
- Database Connections settings
- PROPATH settings
- WebSpeed settings
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. OpenEdge Architect automatically connects to selected databases when you open the project.

To access the Database Connections page, right-click an OpenEdge project name in the Resources view and choose Properties from the context menu. Expand the 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 OpenEdge Architect>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>
<tr>
<td>Connections viewer</td>
<td>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.</td>
</tr>
<tr>
<td>Connection String</td>
<td>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.</td>
</tr>
</tbody>
</table>
**Select All** | Selects all rows.
---|---
**Deselect All** | Deselects all rows.
**Edit** | Starts an **Edit Connection Profile** dialog, where you can change connection parameters.

**Related Tasks**
- Setting project properties

**Related Reference**
- OpenEdge settings
- AppBuilder settings
- Build settings
- Custom settings
- PROPATH settings
- WebSpeed settings
Modules properties page

The Modules properties page allows you to view and configure the module that contains the files that will be published to the AppServer.

**Note:** The Modules properties page property page is present only if the OpenEdge AppServer facet nature is installed on the currently selected project.

You install the OpenEdge AppServer facet when you create a new OpenEdge project and select **Add AppServer module** in the New Project wizard. Existing OpenEdge projects can be migrated to use the AppServer facet in the **Add AppServer support for OpenEdge project** wizard.

To access the Modules page, right-click an AppServer project name in the Resources view and choose **Properties** from the context menu. Expand the OpenEdge node in the tree view, and then click the **Modules node**.

The following controls are available:

<table>
<thead>
<tr>
<th>Module viewer</th>
<th>A tree view that shows the module name and the folders that are selected to be published to the AppServer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modify</td>
<td>When the module name is selected in the tree view, click the Modify button to open the <strong>Modify OpenEdge AppServer module</strong> dialog, which allows you to change the module name and to select the folders that will be published to the AppServer.</td>
</tr>
</tbody>
</table>

**Related Concepts**
- Publishing
- Project support for AppServers
- Terminology related to AppServer support

**Related Tasks**
- Working with AppServer projects

**Related Reference**
- AppServer-related views, pages, and wizards
- AppServer-related preferences
OpenEdge Projects

PROPATH and Source properties page
The PROPATH properties page has two tabs, PROPATH and Source.
To access the PROPATH properties page, right-click an OpenEdge project name in the
Resources view and choose Properties from the context menu. Expand the
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 OpenEdge
Architect>Shared OpenEdge 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:
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.

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.

Add External Library

Adds a procedure library from the file system.

Add Workspace
Directory

Adds a directory from the current workspace.

Add External Directory

Adds a directory from the file system.

Add Standard Paths

Displays a list of common locations that you can choose
to add to your PROPATH setting.

The Source tab allows you to specify the directories in the project where source code
120


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. When you expand a directory in the list, you see two nodes: **Environment** (the runtime environment) and **Build Destination** (where r-code is created). You can specify a unique environment and build destination for each source directory in your project. If there are no entries shown in the Source directory viewer (the default condition), the source folder is the root directory of the project. The default destination folder for r-code is the same folder that contains the source file. However, the default can be overridden if a build destination is specified on the Build properties page. **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. |
| **Edit** | 1. If a source directory is selected, **Edit** allows you to change to a different source folder in the project.

2. 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.

3. If **Build Destination** is selected, **Edit** allows you to designate where to put the r-code in the file system. |

**Related Tasks**

- Setting project properties
- Adding resources to a project
Related Reference

- OpenEdge properties page
- Build properties page
- Database connections properties page
- Modules properties page
- WebSpeed properties page
- Configuration variables
WebSpeed properties page

The WebSpeed properties page allows you to define the broker URL and path settings to use with WebSpeed applications. You can ignore this page if you do not use WebSpeed.

To access the WebSpeed page, right-click an OpenEdge project name in the Resources view and choose Properties from the context menu. Expand the OpenEdge node in the tree view, and then click the WebSpeed node.

The following controls are available:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Broker URL</strong></td>
<td>The URL of the WebSpeed broker configured for your Web server. Leave this blank if you are using the internal server that is included with OpenEdge Architect. <strong>Note:</strong> The internal Web server included with OpenEdge Architect is functionally limited. The recommended approach is to use an external Web server.</td>
</tr>
<tr>
<td><strong>Virtual path</strong></td>
<td>A name used to refer to a location containing required objects or services. This value is ignored if there is an entry in the Broker URL field.</td>
</tr>
<tr>
<td><strong>Real path</strong></td>
<td>The actual location corresponding to the virtual path. This value is ignored if there is an entry in the Broker URL field.</td>
</tr>
<tr>
<td><strong>Add</strong></td>
<td>Adds an entry to the end of the list. Click and enter a path value.</td>
</tr>
<tr>
<td><strong>Edit</strong></td>
<td>Changes the value of a current entry. Select the entry, click Edit, and make the appropriate changes.</td>
</tr>
<tr>
<td><strong>Remove</strong></td>
<td>Deletes an entry from the list. Select the entry and click Remove.</td>
</tr>
<tr>
<td><strong>Up, Down</strong></td>
<td>Changes the position in the list of a current entry and thus change the search order. Select the entry and click Up or Down.</td>
</tr>
</tbody>
</table>

**Related Tasks**

Setting project properties

**Related Reference**

Configuration variables
OpenEdge settings
AppBuilder settings
Build settings
Custom settings
Database Connections settings
PROPATH settings
Introducing AppServer support in OpenEdge Architect

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.

AppServer support is now available directly from OpenEdge Architect and comprises two main areas of functionality:

- **AppServer runtime support** — Lets you control certain AppServer operations, such as starting or stopping AppServers, as well as changing some AppServer properties, such as working directory, PROPATH, and database connections. For a client to access the AppServer from OpenEdge Architect, the client must know where the AppServer is installed, which indicates where the required .jar files and .dll files are located. OpenEdge Architect automatically creates a runtime for its current installation the first time it runs. You can choose, however, whether to create a new runtime for the current version or other installations where a server product is installed.

- **Project functionality** — Lets you publish your application code to a local AppServer for testing.

The inclusion of runtime and project support for AppServers in OpenEdge Architect relies on the Eclipse Web Standard Tools (WST).

With the new AppServer support, you are able to:

- 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.

  Progress Explorer remains available, as does OpenEdge Explorer. You can access OpenEdge Explorer 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 define multiple AppServer connections, enabling you to create and store alternate configurations for each AppServer in your environment.
- Monitor AppServer broker and server status details through a server monitor view.

**Debugging**

OpenEdge Architect does not support debugging of AppServer instances. To debug an AppServer, you must use the standalone OpenEdge Debugger. For more information, see AppServer debugging.

**Related Concepts**

The AppServer perspective
Project support for AppServers
**Related Tasks**
- Defining an AppServer connection
- Working with AppServer connections and agents
- Working with AppServer projects
- Publishing AppServer code to a server for testing

**Related Reference**
- Add AppServer support for OpenEdge project wizard
- OpenEdge Architect Server preferences
- OpenEdge Server Monitor view
Concepts

This section includes the following topics:

- The AppServer perspective
- OpenEdge Architect Server preferences
- AppServer launch configuration
- Project support for AppServers
- Publishing
- Terminology related to AppServer support
The AppServer perspective

The AppServer perspective includes the various views that you use to perform related activities in OpenEdge Architect. The perspective is based on the OpenEdge Editor perspective but also includes the Servers view and the OpenEdge Server Monitor view.

By default, the AppServer perspective includes the following views:

- **Resources** - 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. 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 AppServers.
- **OpenEdge Server Monitor** — A status view for an AppServer broker and its agents.

**Related Concepts**

- Publishing
- Project support for AppServers

**Related Tasks**

- Accessing the AppServer perspective
- Working with AppServer connections and agents
- Viewing AppServer broker and agent status
- Working with AppServer projects
- Publishing AppServer code to a server for testing

**Related Reference**

- AppServer-related views, pages, and wizards
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 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.

**Related Concept**

Launch configurations for running and debugging programs

**Related Tasks**

- Defining a launch configuration
- Setting launch configuration preferences

**Related Reference**

Launch configuration settings
Publishing

You can use the Eclipse publishing functionality to copy files (projects, resource files, and server configurations) from an AppServer project module in the Workspace environment to a local 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 a publish to the server:

- Changing a file in a module
- Removing a file from a module
- Adding a new directory to 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
- 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, and you cannot publish to a remote server.

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 that 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.

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*.

**Related Concept**

Introducing AppServer support in OpenEdge Architect

**Related Tasks**

Publishing AppServer code to a server for testing
Using the Clean option before republishing

**Related Reference**

Server Editor
AppServer-related preferences
Project support for AppServers

Most operations that you perform with OpenEdge Architect take place within the context of a project. You can create multiple projects and maintain them in your workspace concurrently.

You can create an AppServer project that contains additional natures and an AppServer module. For details, see Creating an AppServer project.

Related Concept
- Introducing AppServer support in OpenEdge Architect

Related Task
- Working with AppServer projects
- Introducing OpenEdge Projects
# Terminology related to AppServer support

The following terms describe various components that apply to AppServer support in OpenEdge Architect:

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Server</strong></td>
<td>An instance of a server runtime. You can define many servers for a given runtime. For AppServer support in OpenEdge Architect, a server is used to connect to the AdminServer and provides connection information, broker name, and publishing options for a particular AppServer.</td>
</tr>
<tr>
<td><strong>Server configuration</strong></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><strong>Server Runtimes (or runtimes)</strong></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 the AdminServer and provide AppServer functionality.</td>
</tr>
<tr>
<td><strong>Modules</strong></td>
<td>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><strong>Facets</strong></td>
<td>Used as a way of adding 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 this project as a specific project type.</td>
</tr>
</tbody>
</table>

For more information, see the details in the [Web Tools Platform Guide](#).

**Related Concepts**

- The AppServer perspective
- Project support for AppServers
**Related Tasks**

- Defining an AppServer connection
- Defining the OpenEdge AVM runtime
- Working with AppServer projects

**Related Reference**

- Server Editor
- New Server wizard
- Add AppServer support for OpenEdge project wizard
Tasks

For general information about working with the AppServer in OpenEdge Architect, see:

- Accessing the AppServer perspective
- Setting OpenEdge Architect Server preferences
- Defining an AppServer connection
- Defining the OpenEdge AVM runtime
- Viewing or modifying AppServer properties
- Accessing OpenEdge Explorer

For details about working with AppServer connections and agents, see:

- Starting and stopping the AppServer
- Adding AppServer agents
- Trimming AppServer agents
- Renaming an AppServer connection
- Deleting an AppServer connection

For details about keeping track of AppServer broker and agent activity, see:

- Viewing AppServer broker agent status
- Displaying the AppServer broker and server log files

For details about creating AppServer projects and publishing application code for testing, see:

- Creating an AppServer project
- Configuring AppServer project module properties
- Associating modules with servers
- Using facets in an AppServer project
- Adding or removing AppServer project modules
- Migrating existing projects support AppServer modules
- Publishing AppServer code to a server for testing
- Using the Clean option before republishing
Accessing the AppServer perspective

To open the AppServer perspective:

1. Choose one:
   - Click Open Perspective on the Workbench toolbar.
   - Select Window > Open Perspective.

2. Select OpenEdge AppServer, if it is listed. If it is not listed, choose Other. Choose OpenEdge AppServer from the Open Perspective dialog, and then click OK.

To view the AppServer perspective if it is open but not in view, click OpenEdge AppServer perspective.

Related Concept

The AppServer perspective

Related Tasks

Working with AppServer connections and agents
Viewing AppServer broker and agent status
Working with AppServer projects

Related Reference

OpenEdge Server Monitor view
Server Editor

For more information on using perspectives, toolbars, and menu bars, see the Workbench User Guide in the Eclipse help system.
Setting OpenEdge Architect Server preferences

You can set several OpenEdge Architect Server preferences. Setting preferences allows you to establish certain aspects of AppServer project functionality.

To set the preferences:

1. Choose one:
   - From the OpenEdge Architect menu bar, select Window > Preferences. The Preferences page appears. Select OpenEdge Architect > Server.
   - From the OpenEdge Server Monitor view, click the drop-down menu on the view toolbar and select Preferences.

2. Set each of the following preferences:
   - OpenEdge Explorer 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
   - Publish filters
   - Filter
   - Server status refresh interval
   - Enable AppServer State tab

   (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:
   - 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.
   - To remove a filter, select a filter and click Remove.

4. When you finish, click Apply.

5. Click OK.

Related Tasks
Publishing AppServer code to a server for testing
Accessing OpenEdge Explorer

Related Reference
OpenEdge Server Monitor view
Defining an AppServer connection

Use the New Server wizard to define a connection to the AdminServer and select an AppServer broker to manage. Note that you cannot create or define a new AppServer broker from within OpenEdge Architect. To create a new AppServer broker, use OpenEdge Explorer or Progress 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 the AppServer connection:

1. From the Servers view in the AppServer perspective, right-click and choose New > Server from the Context menu. The Define a New Server dialog appears, allowing you to choose the type of server you want to create.

   The server host name localhost is prefilled. You must use localhost as the name for an AppServer connection.

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.

   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.

   Type the runtime environment name and the OpenEdge version number in the fields provided; then click Finish.

   To set up the runtime environment, click Configure runtime environments and follow the steps provided in Defining the OpenEdge AVM runtime.

4. Click Next.

5. Provide the following information:
   - Server name — The name of the AppServer.
   - Port number — The port number for the connection. You can accept the port number that appears in the field, or you can type a new one.
   - User — The name of the user. You can accept the user name that appears in the field, or you can type a new one.
   - Password — The user’s password (optional).
   - Broker — The name of the broker. You can type the name of the broker in the field provided, or you can click Select to choose from a list of available brokers. (Remember that you cannot create a new AppServer broker in OpenEdge Architect; to create a broker, use OpenEdge Explorer or Progress Explorer.)
Note that the name in the Host field is prefilled as localhost.

6. Click Next to define the publishing rules.

7. Configure the server publish location by choosing one of these options:
   - Select the Use AppServer working directory option (the default).
   - Select the Use custom publish directory option, and provide the path to its location in the Publish directory field. You can also click Browse to find and choose a directory on the local machine.

8. Select or deselect the Publish source code and/or the Publish r-code options. (These options are selected by default.)

9. Click Next. The Add and Remove Projects dialog appears, showing the available project modules.

10. To configure an available project module, select it in the Available projects list and click Add. To configure all the available projects, click Add All.

    You can also remove a project module from the Configured projects list by selecting it and clicking Remove (or select more than one project module and click Remove All).

11. Click Finish.

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.

**Related Concepts**

- Publishing
- Project support for AppServers
- Terminology related to AppServer support

**Related Tasks**

- Accessing OpenEdge Explorer
- Defining the OpenEdge AVM runtime

**Related Reference**

- New Server wizard
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.

OpenEdge Architect automatically creates a runtime for the current installation the first time you run OpenEdge Architect. 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.

To access the list of installed runtime environments:

1. From the OpenEdge Architect menu bar, select Window>Preferences. The Preferences page appears.
2. Select Server/Runtime Environments. The Server Runtime Environments page appears, showing the list of all available OpenEdge runtimes.

You can now do the following:

- Use, modify, remove, or search for an installed OpenEdge runtime environment
- Add an OpenEdge installed runtime environment
- Add a third-party runtime environment

Using, modifying, removing, or searching 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 on 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 on the Preferences page.
Adding an OpenEdge installed runtime environment

From the Server Runtime Environments preferences page, you can add an OpenEdge installed runtime environment.

2. Type OpenEdge in the Select the type of runtime environment field to filter the results, or select Progress Software Corporation>OpenEdge as the runtime environment from the list provided.

(Note that the Create a new local server option is not supported).
3. Click Next.
4. Type the Runtime environment name and OpenEdge version in the fields provided.
5. Click Finish.

Adding a third-party runtime environment

From the Server Runtime Environments Preferences page, you can 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.

For more information, see the Web Tools Project help.

Related Concept

Introducing AppServer support in OpenEdge Architect

Related Reference

Server Runtime Environments preferences
Viewing or modifying AppServer properties

By using the Server Editor, you can view or modify the AppServer properties that define the connection to the AdminServer and the AppServer broker.

To view or modify the AppServer properties:

1. Open the Server Editor in either of the following ways from the **Servers** view in the AppServer perspective:
   - 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
   - OpenEdge Explorer
3. Make any changes, and click **OK**.
Accessing OpenEdge Explorer

You can access OpenEdge Explorer from several different perspectives, including the AppServer. (It might be necessary for you to reset the perspectives in existing workspaces to see new view and perspective shortcuts.)

From the OpenEdge Architect 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 and a server is currently selected in the **Servers** view, that server's OpenEdge Explorer URL is used to open the Web browser. If a server is not selected in the **Servers** view, the URL used is the one defined for OpenEdge Explorer in the Preferences page.

**Related Reference**

OpenEdge Architect Server preferences
Working with AppServer connections and agents

This section includes the following topics:

- Starting and stopping the AppServer
- Adding AppServer agents
- Trimming AppServer agents
- Renaming an AppServer connection
- Deleting an AppServer connection

Related Concept
The AppServer perspective

Related Task
Defining an AppServer connection

Related Reference
OpenEdge Server Monitor view
Server Editor
Starting and stopping the AppServer

You can start or stop the AppServer from the Servers view in the AppServer perspective.

1. From the Servers view, select an AppServer.
2. Choose one:
   - To start or restart the server, either 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 click Start the server in debug mode on the view toolbar, or right-click the server name and select Debug.
   - To stop the server, either click Stop on the view toolbar, or right-click the server name and select Stop.

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.

Related Task
Viewing AppServer broker and agent status

Related Reference
OpenEdge Server Monitor view
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. (You can add agents up to the maximum number that your license allows.)

Likewise, if there is a reduction in the number of requests, you can reduce (trim) the number of running ApAppServer 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 AppServer Agents**. The **Add Agents** dialog appears.
3. Type the number of agents you want to add in the **Count** field.
4. Click **OK**.

**Related Task**

Trimming AppServer agents

**Related Reference**

Add/Trim Agents dialog
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 AppServer Agents.
   - To specify the number of agents you want trimmed, select Trim AppServer 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.

Related Tasks

Adding AppServer agents
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.

Related Tasks
- Defining an AppServer connection
- Deleting an AppServer connection
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. You can select the Stop server(s) before deleting option, if you want to stop an active connection before you delete it.
4. Click OK.

Related Tasks

Defining an AppServer connection
Renaming an AppServer connection
Viewing AppServer status

You can view AppServer status in the following ways:

- By looking at the status of brokers and agents
- By looking at the server and broker log files
Viewing AppServer broker and agent status

You can view the status of an AppServer broker and agents in the **OpenEdge Server Monitor** view.

Open the **OpenEdge Server Monitor view** in either of the following ways:

- From the **AppServer perspective**, click the **OpenEdge Server Monitor** view tab.
- From the OpenEdge Architect menu bar, select **Window > Show View > OpenEdge Server Monitor**.

The view comprises three tabs:

- **Agent Summary** — A summary of status information for the selected AppServer broker and agents.
- **Agent Detail** — A detailed view of the status of each AppServer agent.
- **AppServer State** — A summary of the PROPATH, database, and persistent procedures for a single AppServer agent.

**Automatically refreshing the view**

You can opt to automatically refresh the OpenEdge Server Monitor view. From the view, click **Automatically refresh server status** in the view toolbar.

**Refreshing the server status**

To refresh the server status, click **Refresh server status** in the view toolbar.

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 OpenEdge Architect or by a different OpenEdge Architect session, the Console view is unavailable.)

You can open the Console view to display the broker log file or the server log file in either of the following ways:

- From the Servers view, select the server or broker and right-click. Then choose Show In > Console.
- From the OpenEdge Architect menu bar, select Window > Show View > Console.

Related Tasks

Starting and stopping the AppServer
Viewing AppServer broker and agent status
Working with AppServer projects

This section includes the following topics:

- Creating an AppServer project
- Configuring AppServer project module properties
- Associating modules with servers
- Using facets in an AppServer project
- Adding or removing AppServer project modules
- Migrating existing projects to support AppServer modules
Creating an AppServer project

Using the New Project wizard, you can create an AppServer project that contains an AppServer module. The module contains the files that you plan to publish.

To create a new project with an AppServer module:

1. Select File>New>OpenEdge Project from the OpenEdge Architect menu bar. The New OpenEdge Project wizard opens.
2. Type the project name in the Project name field.
3. Choose the project location. Use default location is selected by default, but you can deselect it and type a different location, or click Browse to find another location.
4. Choose whether to use a project-specific AVM or a shared OpenEdge AVM. Select Use TTY for runtime if you want to enable that option.

Note: If you are publishing r-code, choose the Use TTY for runtime option for the project to ensure that code containing UI elements (such as display statements) compiles properly. Otherwise, you can publish only source code.

5. Choose the project layout:
   - Use project root directory for source and r-code.
   - Use separate source and r-code directories. If you select this option, select the Source directory and R-code directory options, and type the name of each directory in the field provided.
6. Select the Add AppServer module option.
7. Click Next. Type the module name and the AppServer source folder name in the fields provided. To avoid confusion, be sure to give each module a unique name.

   The AppServer source folder is automatically included as part of the project when you finish the project creation. The folder will also be added to the PROPATH for the project as a source directory, and the default r-code destination will be that of the project.

   You add to this source folder any files that must be published to the AppServer.
8. Select one or more servers that you want to publish the AppServer module. To publish the module to the server after you finish creating the project, click the Publish changes immediately option.
9. Click Next. Specify the PROPATH to use for the project.
10. Click Next. Specify the database connection(s) to use for the project.
11. Click Finish.

You can add other module content resources through the AppServer module configuration property page after the new project wizard is finished.

Related Concept

Publishing
**Related Tasks**

- Defining the OpenEdge AVM runtime
- Publishing AppServer code to a server for testing

**Related Reference**

- Introducing OpenEdge Projects
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 Resources view, and then select File>Properties from the OpenEdge Architect menu bar.
   - Select the AppServer project in the Resources view, and then right-click. From the Context menu, select Properties.

2. Select OpenEdge>Modules. The Modules Properties page appears.

   By default, the AppServerContent 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 and select which folders you want to add to or remove from the module.

4. Click Apply. (You can alternatively click Restore Defaults to return the settings to their original values.)

5. Click OK when you finish configuring folders for the module.

Related Reference

Modules Properties page
Web Tools Platform User Guide
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 projects from the Context menu. The Add and Remove Projects 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 projects list, and then click Add to move the module to the configured Projects list.

If you want to move all the available project modules, click Add All. To remove an individual project module, select it in the configured projects list and click Remove. To remove all project modules from the list of configured projects, click Remove All.

You can also select the option to publish changes immediately upon the startup of the server.

**Related Concept**
- Project support for AppServers

**Related Task**
- Adding or removing AppServer project modules

**Related Reference**
- Add and Remove Projects page
Using facets in an AppServer project

You can create an AppServer project that contains additional natures, an AppServer module, and project facets. A nature (which allows the WTP server framework to add a module to a project) is hidden from your view and typically works behind the scenes, whereas a project facet is intended to be visible to and used by you. It is the installation of the facet that creates the module.

For more general information about facets, see the Web Tools Platform User Guide.

To use AppServer project facets:

1. Open the project's properties. You can do this in several ways, including:
   - Select the project in the Resources view, and then select File > Properties from the OpenEdge Architect menu bar.
   - Select the project in the Resources view, and then right-click. From the Context menu, select Properties.

2. Select Project Facets from the list of properties shown in the left pane.

   **Note:** The Project Facets option is listed in the properties only if the facet nature has been installed on the project. Existing OpenEdge projects do not have project facets unless you migrate them to use facets.

   The Project Facets page appears, with the OpenEdge and the OpenEdge AppServer facets (among others) listed and selected.

3. If you want to remove the facet and the module definition file, deselect the OpenEdge AppServer facet and click Apply. You cannot remove the OpenEdge facet.

   To reinstall the OpenEdge AppServer project facet, select it in the Project Facets page. The Further configuration available link appears near the bottom of the page.

4. Click the link. The Modify Faceted Project dialog appears for the OpenEdge AppServer project.

5. Type the module name, and select the folders you want published.

6. Click OK.

**Related Task**

Associating modules with AppServers
Adding or removing AppServer project modules

You can add and remove AppServer project modules that are configured on the server.

To add or remove AppServer project modules:

1. From the **Servers** view in the AppServer perspective, select the server where the project resides.
2. Right-click the server, and select **Add and Remove Projects** from the **Context** menu. The **Add and Remove Projects** dialog appears.
3. To add a project module, select it in the **Available projects** list on the left, and click **Add**. To add all modules for all available projects, click **Add All**. The project module or project modules are added to the **Configured projects** list on the right.

   To remove a project module, select it in the **Configured projects** list on the right, and click **Remove**.

   To remove all configured project modules, click **Remove All**. The project module or project modules are moved to the **Available projects** list on the left.
4. When you are done, click **Finish** to save the changes.

*Related Concept*

The AppServer perspective

*Related Reference*

Add and Remove Projects page
Migrating existing projects to support AppServer modules

Existing OpenEdge projects do not have the natures and facet information necessary to define an AppServer module in a project. To enable a project for AppServer functionality, use the Migration wizard.

For a single project that you select, the Migration wizard adds the required nature and facet information as well as prompts for the AppServer Content directory. The wizard adds the AppServer Content directory to the PROPATH and prompts you to check out or make readable any required files. The migrated project becomes the equivalent of an AppServer project that you create by using the New Project wizard.

To migrate an existing OpenEdge project to support the AppServer:

1. From the OpenEdge Architect menu bar, select File> New > Other. The Select a wizard page opens.
2. Select OpenEdge> Add AppServer support for OpenEdge project, and click Next.
3. Select the project you want to enable for AppServer support, and click Next.
4. Specify the folders you want to publish. You can choose the project folder to publish all the folders within the project, or select individual folders within the project.
5. Do either of the following:
   - To save your selections and exit the wizard, click Finish.
   - To choose one or more AppServers to publish the module, click Next. A list of defined AppServers appears. Select the servers that you want to publish the module. If you want to publish the changes immediately, select that option. When you are done, click Finish.

The project now contains a new AppServer module; and additional, required natures and facet identifiers are added to the project. To access the new property pages for the facets and the module, open the Project Properties dialog.

If you chose to associate the new module with an existing server, the server displays the project as a module in the Servers view.

Related Task

Working with AppServer projects

Related Reference

Add AppServer support for OpenEdge project wizard
Modules properties page
Publishing AppServer code to a server for testing

Publishing code to a server so you can test it involves the following overall steps:

1. Associating modules with servers
2. Publishing to a local server
3. Removing published content from the server

Related Concept

Publishing
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.

To use the Clean option:

1. From the **Servers** view, select a server and right-click. The Context menu appears.
2. Select **Clean**, and then click **OK** to confirm.

**Related Concept**

- Publishing

**Related Task**

- Publishing AppServer code to a server for testing
Reference

This section includes the following topics:

AppServer-related views, pages, and wizards
AppServer-related preferences
AppServer-related views, pages, and wizards

AppServer-related views, pages, and wizards include the following:

- OpenEdge Server Monitor view
- New Server wizard
- Add AppServer support for OpenEdge project wizard
- Server Editor
- Add/Trim Agents dialog
- Broker Selection dialog
OpenEdge Server Monitor view

The OpenEdge Server Monitor view provides a status summary for the AppServer broker and agents. You can access the view by either:

- Opening the AppServer perspective and clicking the OpenEdge Server Monitor view tab.
- Choosing Window>Show View>OpenEdge Server Monitor from the OpenEdge Architect menu bar.

The view provides details about the following:

<table>
<thead>
<tr>
<th>Broker name</th>
<th>The name of the AppServer broker.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating mode</td>
<td>The operating mode: State-reset, State-aware, Stateless, and State-free.</td>
</tr>
<tr>
<td>Broker status</td>
<td>The broker status: starting, active, stopping.</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>
</tbody>
</table>

The view provides additional details on three status tabs and also includes a toolbar and a menu bar that you can use to perform certain AppServer tasks.

Status tabs

The status information in the view is available from the following three tabs:

- Agent Summary
- Agent Detail
- AppServer State

Agent Summary tab

The following summary details related to broker and agent status appear in the Agent Summary tab:

<table>
<thead>
<tr>
<th>Active agents</th>
<th>The number of active agents, including all those busy, locked, and available</th>
</tr>
</thead>
<tbody>
<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>
</tbody>
</table>
A detailed view of the status of each AppServer agent appears in the Agent Detail tab:

<table>
<thead>
<tr>
<th>Active clients</th>
<th>The number of active client requests (current, peak)</th>
</tr>
</thead>
<tbody>
<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>

Agent Detail tab

A detailed view of the status of each AppServer agent appears in the Agent Detail tab:

<table>
<thead>
<tr>
<th>PID</th>
<th>The operating machine process ID of the agent executable</th>
</tr>
</thead>
<tbody>
<tr>
<td>State</td>
<td>The agent status: busy, locked, available, stopping, or starting</td>
</tr>
<tr>
<td>Port</td>
<td>The port number that the agent is using to communicate with clients</td>
</tr>
<tr>
<td>Requests</td>
<td>The number of requests processed by this AppServer agent</td>
</tr>
<tr>
<td>Received</td>
<td>The number of received requests processed by this AppServer agent</td>
</tr>
<tr>
<td>Sent</td>
<td>The number of responses sent by this AppServer agent</td>
</tr>
<tr>
<td>Started</td>
<td>The date/time this agent was started</td>
</tr>
<tr>
<td>Last Change</td>
<td>The date/time this agent last changed its status or processed a request</td>
</tr>
</tbody>
</table>

AppServer State tab

If you have enabled the AppServer State tab in the AppServer preferences, details appear on the tab when you open it. If you have not enabled the AppServer State tab, no details appear. Instead, a message informs you that the tab is disabled and points you to the AppServer preferences page, where you can enable the option.

Note that AppServer State details are not available for pre-OpenEdge 10.1A AppServers.

The following information for a single AppServer agent appears in the AppServer State tab:
tab when you have enabled it:

<table>
<thead>
<tr>
<th>Connection bound req</th>
<th>Value of the AppServer SESSION:SERVER-CONNECTION-BOUND-REQUEST attribute (LOGICAL), which indicates if the AppServer is requesting a bound connection with the client</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection bound</td>
<td>Value of the AppServer SESSION:SERVER-CONNECTION-BOUND attribute (LOGICAL), which indicates if the AppServer is currently in a bound connection with the client</td>
</tr>
<tr>
<td>PROPATH</td>
<td>The search path that agents use to locate ABL procedures that they execute</td>
</tr>
<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 that the details shown in the **AppServer 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**.

**OpenEdge Server Monitor view toolbar**

You can use the buttons on the OpenEdge Server Monitor view toolbar to accomplish the following tasks:

<p>| ![Pin view] | Pin the view to display only the current AppServer broker. If you pin the view to the current server and then choose another server, a new copy of the OpenEdge Server Monitor view is created and the selected broker appears in the new view. You can repeat this action to see as many AppServer views as you want at one time. The current view continues to display the currently selected AppServer broker. |
| ![Refresh] | Automatically refresh the server status at a predetermined interval. |
| ![Refresh] | Refresh the server status. |
| ![Add]     | Add one or more agents. |
| ![Trim]    | Trim one or more agents. |</p>
<table>
<thead>
<tr>
<th></th>
<th>Trim all agents.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Open a Web browser by using the currently selected server and start OpenEdge Explorer. If there is no server selected, the OpenEdge Explorer option is disabled. For more information, see Accessing OpenEdge Explorer.</td>
</tr>
<tr>
<td></td>
<td>Access a list of available brokers and AppServers available to monitor, whether they are started or not. If there are no other servers to monitor, the word <strong>unavailable</strong> appears in the list. Also provides access to the OpenEdge Architect Server preferences.</td>
</tr>
</tbody>
</table>
New Server wizard

Use the Eclipse New Server wizard to define an AppServer connection from OpenEdge Architect. You access the wizard by selecting File>OpenEdge>AppServer. The following pages appear sequentially in the wizard:

Define a New Server
Define a new AppServer broker
Define publishing rules
Add and Remove Projects

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>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server's host name</td>
<td>For the AppServer, the host name must be localhost.</td>
</tr>
<tr>
<td>Select the server type</td>
<td>Type AppServer in the filter field provided, or select OpenEdge&gt;AppServer.</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 AppServer connection:

<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</td>
<td>The host name, which is prefilled.</td>
</tr>
<tr>
<td>Port number</td>
<td>The port number for the AdminServer connection.</td>
</tr>
<tr>
<td>User</td>
<td>The name of the user.</td>
</tr>
<tr>
<td>Password</td>
<td>The user's password (optional).</td>
</tr>
</tbody>
</table>
Define publishing rules page

From the Define publishing rules page, you can specify a publish location for the AppServer connection and identify the file types to be published.

<table>
<thead>
<tr>
<th>Use AppServer working directory (the default) or Use custom publish directory</th>
<th>Select the server publish location. If you select the custom directory, you provide the path to its location in the Publish directory field. You can also click Browse to select another directory on the local machine.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publish source code</td>
<td>Select to publish the source code. This option is selected by default.</td>
</tr>
<tr>
<td>Publish r-code</td>
<td>Select to publish the r-code. This option is selected by default.</td>
</tr>
<tr>
<td>Back</td>
<td>Click to move back to the Define a new AppServer broker page, the previous page in the wizard.</td>
</tr>
<tr>
<td>Next</td>
<td>Click to move to the Add and Remove Projects page, the next page in the wizard.</td>
</tr>
</tbody>
</table>

Add and Remove Projects page

From the Eclipse Add and Remove Projects page, you can add or remove project modules that are being configured on the server.

<p>| Available projects | A list of project modules available for configuration. |
| Configured projects | A list of configured project modules. |
| Add | Select an available project module and click Add to move it to the list of configured project modules. |</p>
<table>
<thead>
<tr>
<th>Remove</th>
<th>Select a configured project module and click <strong>Remove</strong> to move it to the list of available project modules.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add All</td>
<td>Click to move all available project modules to the list of configured project modules.</td>
</tr>
<tr>
<td>Remove All</td>
<td>Click to move all configured project modules to the list of available project modules.</td>
</tr>
<tr>
<td>Back</td>
<td>Click to move back to the <strong>Define publishing rules</strong> page, the previous page in the wizard.</td>
</tr>
<tr>
<td>Finish</td>
<td>End adding and removing project modules.</td>
</tr>
</tbody>
</table>

**Related Concept**

Publishing

**Related Tasks**

Defining an AppServer connection
Defining the OpenEdge AVM Runtime
Viewing AppServer status
Publishing AppServer code to a server for testing
Accessing OpenEdge Explorer

**Related Reference**

Server Runtime Environments preferences
Add AppServer support for OpenEdge project wizard

The Add AppServer support for OpenEdge project wizard allows you to select an existing OpenEdge project and migrate it, enabling it for AppServer support. You can access the wizard by selecting File > New > Other > OpenEdge > Add AppServer support for OpenEdge project.

These pages appear in the following order in the wizard:

- Enable a project for AppServer functionality
- Define the AppServer module
- Select AppServers

The wizard also allows you the option of publishing changes immediately.

**Enable a project for AppServer functionality**

Select the project in which you want to enable AppServer support.

**Define the AppServer module**

Provide the name of the AppServer module, and select which project folders you want to publish.

**Select AppServers**

Select the server or servers to which you want to publish the project’s AppServer module.

**Related Tasks**

- Migrating existing OpenEdge projects to support AppServer modules
- Configuring AppServer project module properties
- Publishing AppServer code to a server for testing
Server Editor

You use the Server Editor to view or modify the server properties that define the connection to the AdminServer and broker. From the Servers view in the AppServer 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 the AdminServer.
- **Publishing** — Specifies when to publish.
- **Timeouts** — Specifies the time limit to complete server operations (Start and Stop).
- **Publish Location** — Specifies the server publish directory.
- **OpenEdge Explorer** — Specifies the OpenEdge Explorer URL for the server.

### General Information

You can modify the following General Information properties in the Server Editor:

<table>
<thead>
<tr>
<th>Server name</th>
<th>The name of the AppServer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host name</td>
<td>The name of the host machine. In the case of the AppServer, the host name is localhost.</td>
</tr>
<tr>
<td>Runtime Environment</td>
<td>The current runtime environment. You can click the drop-down menu or the Runtime Environment 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 Connection properties in the Server Editor:

<table>
<thead>
<tr>
<th>Port</th>
<th>The port where the AdminServer runs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>User name</td>
<td>The user name. The default is the current user.</td>
</tr>
<tr>
<td>Password</td>
<td>The password associated with the specified username.</td>
</tr>
<tr>
<td>Broker name</td>
<td>The AppServer broker name. You can type the name of a different broker, or you can click Broker name to choose another broker for the AppServer.</td>
</tr>
</tbody>
</table>
**Publishing**

You can modify the publishing settings to choose either of the following options:

- Never publish automatically
- Automatically publish when resources change

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 <strong>Publish directory</strong> 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>
</tbody>
</table>

**OpenEdge Explorer**

You can specify whether to use the default OpenEdge Explorer URL or provide a custom URL. The default URL is:

http://localhost:9090/

The URL must be valid.

For more information about starting OpenEdge Explorer, see **Accessing OpenEdge Explorer**.
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.

You can access the dialog from the Servers view. Select an AppServer broker, and then select Add AppServer Agents from the Context menu. Type the number of agents you want to add in the Count field, and then either click OK to add the agents or click Cancel to keep the number of agents as it is.

Related Task

Adding AppServer agents
Broker Selection dialog

The Broker Selection dialog provides a list of all AppServer brokers that have previously been created and is used in both the New Server wizard and in the Connection section of the Server Editor.

In the New Server wizard, the dialog appears when you opt to click Select (rather than type the name of a broker) for the Broker field on the Define a New AppServer broker page. Select one of the brokers to complete the wizard.

In the Connection section of the Server Editor, the dialog appears if you click the Broker name link rather than type the name of the broker in the field provided.

The dialog does not allow you to create a new AppServer broker. To create a new AppServer broker, use OpenEdge Explorer or Progress Explorer.

Related Concept
Project support for AppServers

Related Tasks
Defining an AppServer connection
Accessing OpenEdge Explorer
AppServer-related preferences

You can set AppServer-related preferences in the following pages:

Server Runtime Environments preferences page
OpenEdge Architect Server preferences page
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/Runtime Environments.

You can add, edit, or remove server runtime environments by using the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>To select a new OpenEdge AVM runtime, click Add.</td>
</tr>
<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>

**Related Concept**

Introducing AppServer support in OpenEdge Architect

**Related Task**

Defining the OpenEdge AVM runtime
OpenEdge Architect Server preferences

You can set several OpenEdge Architect Server preferences related to publishing and to the OpenEdge Server Monitor view.

You can access the OpenEdge Architect Server preferences page in several different ways, such as:

- From the OpenEdge Architect menu bar, by choosing Window > Preferences > OpenEdge Architect > Server.
- From the OpenEdge Server Monitor view, by clicking the dropdown menu on the view toolbar and selecting Preferences.

You can set these Server preferences:

<table>
<thead>
<tr>
<th>OpenEdge Explorer 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. This option is enabled by default.</td>
</tr>
<tr>
<td>Remove deleted files and folders on publish</td>
<td>Select this option to remove all files and folders before starting the clean operation. This option is disabled by default.</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. This option is disabled by default.</td>
</tr>
<tr>
<td>Publish empty folders</td>
<td>Select this option if you want to publish folders even if they are empty. This option is disabled by default.</td>
</tr>
<tr>
<td>Publish filters</td>
<td>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.</td>
</tr>
<tr>
<td>Filter</td>
<td>To add a filter, click Add (next to the Publish Filters table) and then type the filter expression in this field. Then click Update.</td>
</tr>
<tr>
<td>Server status refresh interval (in seconds)</td>
<td>Specify the refresh delay interval for server status updates. The default is 15 seconds.</td>
</tr>
<tr>
<td>Enable AppServer State tab</td>
<td>Select this option to enable updating of the server session state page.</td>
</tr>
<tr>
<td>Restore Defaults</td>
<td>Click to restore the original set of publish filters and all other preference options on this page.</td>
</tr>
<tr>
<td><strong>Apply</strong></td>
<td>Apply the preference selections.</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------------------</td>
</tr>
</tbody>
</table>

Introducing the ABL Editor

The ABL Editor is the standard Eclipse text editor customized for working with ABL code. The ABL Editor inherits behavior from the preferences you set for all text editors (Preferences>General>Editors). You can then further customize its behavior with OpenEdge-specific preferences (Preferences>OpenEdge Architect>Editor).

Note: For information on the Eclipse text editor preferences, see the Workbench User Guide, starting with Editors.

The ABL Editor supports many features to make coding easier, including:

- ABL statement completion
- Dragging-and-dropping names from associated views
- Wizards for creating new files and adding code blocks to files
- Macros
- Automatically formatting code

Related Concepts
OpenEdge Editor perspective views

Related Tasks
Setting Editor options
Using basic ABL Editor functions
Writing ABL code
Writing object-oriented ABL code
Running ABL code
Concepts

This section includes the following topics:

- ABL object-oriented extensions
- Code parsing
- Code assistance
- Code folding
- Code annotations
ABL object-oriented extensions

OpenEdge includes significant extensions to the language that allow you to code application objects using an object-oriented development model. Support includes inheritance, polymorphism, delegation, interfaces, and overloading.

These new 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 exactly the same way as in procedural code.

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.

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.

Related Tasks

Writing object-oriented code

Related Reference

ABL Language Reference
## 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. OpenEdge Architect 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 OpenEdge Architect features.

The parser performs a new analysis and updates its in-memory representation whenever a change to the code is followed by a 300-millisecond interval 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 bar of OpenEdge Architect, go to `Window>Preferences>OpenEdge Architect>Editor>Assistance`.

### Related Tasks
- Using code-completion features

### Reference
- Editor Assistance options
Code assistance

The ABL Editor provides several forms of assistance to facilitate writing ABL code:

- Hover help
- Help as you type
- Code completion

Hover help

Hover help displays information about database fields, system-handles, and ABL statements when the cursor is positioned over a particular element in the code.

You can disable this feature in OpenEdge Editor Assistance preferences.

Note: This feature relies on OpenEdge Architect's grammar parsing functionality. The text hover only appears 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
```

System handles also support hover help, as illustrated below:

```
IF ERRCR-STATUS:ERROR THEN DO:
```

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
element. 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 CurrentRowids AS CHARACTER NO-UNDO.
```

Text hover is also applied on attributes and methods:

```
THIS-PROCEDURE: ADD-SUPER-PROCEDURE (h1)
```

**Help as you type**

You can choose to have the ABL Editor display syntax help in a pop-up window whenever
you type an ABL keyword followed by a space. This help is similar to hover help, but
is less detailed.

You can disable this feature in Editor Assistance preferences.
**Code completion**

At any time while editing an ABL source file, you can press **CTRL+SPACE** to display a list of syntax elements that are valid for the context, and you can then select an entry from the list and automatically insert it. Optionally, you can also choose to have the ABL Editor propose completion options automatically (that is, without your pressing **CTRL+SPACE**) when you type a period in a schema reference or a colon in a handle or object reference.

**Related Tasks**

- Setting ABL Editor preferences
- Displaying ABL syntax help
- Using code-completion features
- Checking syntax

**Related Reference**

- Editor Assistance options
Code folding

The Editor supports *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, an icon (minus sign in a blue circle ☹️) displays in the Marker Bar beside the first line of the section. After you fold the code, the first line of the section remains and the icon changes (plus sign in a blue circle ☺️).

The hidden code displays in a text hover when your cursor is over the icon:

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

**Related Tasks**

- Commenting out code
Code 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 reads annotations from your source code and loads them into the Meta Catalog.

You can assign almost any name to an annotation. If you use the same name for an annotation that another content builder uses for the data it adds to the Meta Catalog, your annotations will be sorted with the data from the other content builder.

Related Concepts

ABL annotations

Related Tasks

Adding templates for code annotations
Adding annotations to source files
Adding Sonic ESB annotations

Related Reference

ABL Annotation content builder
ABL Annotation syntax
Tasks

This section includes the following sections:

- Setting Editor options
- Using basic ABL Editor functions
- Writing ABL code
- Writing object-oriented ABL code
Setting Editor options

This section includes the following topics:

- Setting Eclipse editor preferences
- Setting ABL Editor preferences
- Associating custom file extensions with the ABL Editor
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 **Window>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*.

**Note**: You can also choose a font for comparing ABL files.

**Related Tasks**
- Setting ABL Editor preferences
- Associating custom file extensions with the ABL Editor

**Related Reference**
- ABL Editor preferences
Setting ABL Editor preferences

Use preference settings to modify the behavior of the OpenEdge ABL Editor.

1. Select Window>Preferences. The Preferences window appears.
2. Select the OpenEdge Architect>Editor node. The Editor options page appears with settings for indentation style, keyword case, and automatic expansion of keywords and schema references.
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.

Specifying a font for comparing ABL files

Eclipse lets you compare text files by selecting the files in the Resources 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 do so:

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. Click Change.
4. Select a font in the browser and click OK.

Related Tasks

Setting Eclipse editor preferences
Associating custom file extensions with the ABL Editor

Related Reference

ABL Editor preferences
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 Editor. To associate your custom file extensions with the OpenEdge Editor, you need to 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 Editors and click OK.

   Note: The only difference between these editors is the icon used to represent the files that it opens. All the editors are functionally the same.

8. Select OpenEdge Architect>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.

**Related Reference**

Editor Build options
Using basic ABL Editor functions

This section includes the following topics:

- Creating ABL source files
- Adding new templates (macros)
- Adding templates for code annotations
- Adding new templates (macros)
- Automatically formatting code
- Clearing OpenEdge errors
- Comparing files
- Compiling ABL code
- Displaying ABL syntax help
- Displaying the ABL Console view
- Displaying Quick Outline
- Going to a specific line number
- Matching paired elements
- Using code-completion features
- Using templates for easier text entry

**Demos**

- Using the Quick Diff Editing Feature
- Using the History View
Creating ABL source files

OpenEdge Architect provides the following code-generation wizards that serve as the most efficient way to create an ABL source file:

- New ABL Class
- New ABL Include
- New ABL Interface
- New ABL Procedure
- New ABL Structured Include
- New ABL Structured procedure

To invoke one of these wizards, select it in either of the following ways:

- **New menu** - Access this menu either by selecting *File>New* on the workspace menu, or by clicking the down arrow next to *New* on the toolbar. The content of the top-level *New* menu is dynamic, so it may or may not include the wizard you are looking for. If not, select *Other* to open the New Wizard browser. Expand the *OpenEdge* node and then the *Editor* node to find the appropriate wizard.

- **New Wizard browser** - Open this browser by clicking *New* (not the down arrow) on the toolbar. Expand the *OpenEdge* node and then the *Editor* node to find the appropriate wizard.

### Generating code elements

While editing an ABL file, you can use ABL wizards and commands to insert various code elements. You select these options from the *Source menu*.

The *Source* menu is available on the workspace menu bar and on the ABL Editor context (right-click) menu.

**Related Tasks**

- Writing ABL code
- Writing object-oriented ABL code

**Related Reference**

- Code-generation wizards
- Source menu
Adding new templates (macros)

Macros (editor templates) are special character sequences that the Editor expands into ABL phrases.

To add a new macro:

1. Choose Window>Preferences. The Preferences window appears.
2. Choose OpenEdge Architect>Editor>Macros.
3. Click New to display the New Macro dialog.
4. Enter your new macro and click OK.

The Name of the macro is the character sequence that the Editor expands. Choose a unique name for each new macro. If you want to change an existing macro, you can use the Edit Macro dialog.

Related Tasks

Using templates for easier text entry

Related Reference

Editor Templates (Macros) options
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 OpenEdge Architect>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.

**Related Concepts**
- Code annotations

**Related Tasks**
- Adding annotations to source files

**Related Reference**
- Editor Annotations options
- ABL annotation content builder
- ABL annotation syntax
Automatically formatting code

The ABL Editor has a number of features that allow you to apply formatting changes to an open ABL file.

To access formatting features:

1. Open a file in the ABL Editor.
2. Optionally, select a block of code to which you want to apply formatting. If no text is selected, the entire file will be formatted.
3. Select **Source** from the main menu bar or from the context menu in the file.
4. Choose one of these options:

   **Correct Case** - Applies keyword casing. This option applies the keyword casing preference (upper or lower) specified on the Editor preference page (**Window>Preferences>OpenEdge Architect>Editor**).

   **Expand Keywords** - Replaces abbreviated keywords in the current file with their fully spelled equivalents.

   **Correct Indentation** - Indents lines consistently, and also applies tabular formatting if this option is enabled (in OpenEdge Architect Editor preferences).

**Tabular formatting**

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.
```

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 OpenEdge Architect Editor preferences.

**Related Tasks**

- Setting Eclipse editor preferences

**Related Reference**

- Source menu
Editor options
Clearing OpenEdge errors

You can clear any OpenEdge error messages for a particular file from showing up in the Problems view or the Editor's marker bar by choosing OpenEdge>Clear OpenEdge Errors on the Editor's pop-up context menu.

This does not fix the errors. Any unfixed errors are displayed again when you next compile the code.

Related Tasks
  Checking syntax
  Compiling ABL code
  Running ABL code
Comparing files

To compare ABL code files in OpenEdge Architect:

1. Select two files in the Resources 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 they are color-coded in the ABL Editor.

**Note:** You can compare three files when one file is the common ancestor of the other two.

**Related Reference**

Workbench User Guide
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 OpenEdge Architect 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

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 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.
Related Tasks

- Checking syntax
- Clearing OpenEdge errors
- Going to a specific line number
- Running ABL code
- Setting breakpoints while editing

Related Reference

- Editor Build options
Displaying ABL syntax help

You can get several types of help on ABL syntax:

- **Hover help** - Hold the cursor over a keyword or a schema reference and a pop-up window appears with details of its syntax. (To disable this feature, deselect the Show help on schema hover option and/or the Show help on keyword hover option in OpenEdge Editor Assistance preferences.)

  **Note**: This feature relies on OpenEdge Architect’s grammar parsing functionality. The help 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.

- **Help as you type** - Type an ABL keyword followed by a space. A list of phrases valid for the context appears in a pop-up window. (To disable this feature, deselect the Context assistance option in OpenEdge Editor Assistance preferences.)

- **OpenEdge Development: ABL Reference entries** - Highlight a keyword or phrase and press SHIFT+F2 (or choose Keyword Help from the context menu) to display its reference entry in the Keyword Help view in the information area.

Note that the help described here is simply reference text, not code-completion assistance. Code-completion assistance is also available.

**Related Concepts**

- Code assistance

**Related Tasks**

- Setting ABL Editor preferences
- Checking syntax
- Using code-completion features

**Related Reference**

- ABL Language Reference
- Editor Assistance options
Displaying the ABL Console view

The Console is a Workbench view that displays the text output from commands (runtime startup, for example) similar to the output in a command window. It appears in the default OpenEdge Editor perspective.

If the Console view does not appear in your workspace, you can start it by selecting Window>Show View>Console from the main menu bar.

If the Console in your workspace is not the ABL Console, click Select Console in the Console toolbar and select ABL Console from the menu.

Related Concepts
OpenEdge Editor perspective views

Related Tasks
Clearing OpenEdge errors
Displaying the ABL Preprocessor view

The 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. To access the Preprocessor view:

1. Select **Window>Show View>Preprocessor** from the main menu bar.

   It can also be started from the ABL Editor’s context menu by selecting **Show In>Preprocessor**.

2. Click the Link button to refresh the view when focus in the ABL Editor changes to another file.

   By default linking is off.

3. If linking is off, click the Refresh button to update the file displayed in the view.

   Also use the Refresh button to update the view after code changes.

**Related Tasks**

Opening included files
Displaying 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. 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 may be more convenient to work with.

To display Quick Outline:

1. Click in a file that is open in the ABL Editor.
2. Select Navigate>Quick Outline from the main menu, or press CTRL+O.
3. To close Quick Outline, change focus or press ESC.

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.

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.

Related Reference

OpenEdge Editor perspective views
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 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*.

**Related Tasks**

- Setting Eclipse editor preferences
Matching paired elements

The Editor can automatically match paired elements, such as brackets.

When entering brackets and other paired elements, the matching element is automatically inserted in front of the cursor. Positioning the cursor on any bracket highlights the matching bracket in the code sequence. Support for automatic bracketing includes:

- Double and single quotes
- Brackets and parenthesis, "{", ":[", ":"

To enable the insertion of matching elements:

1. Choose **Window>Preferences** to launch the **Preferences** window.
2. Choose **OpenEdge Architect>Editor>Assistance** to display the **Assistance** page.
3. Select the **Auto bracketing** option.

Bracket matching

The standard Eclipse editor provides a visual cueing system for matching elements. Bracket matching is useful for providing visual cues illustrating the location of matched pair elements. Bracket matching includes:

- Literal start, double and single quotes
- Brackets and parenthesis, "{", ":[", ":"
- Code blocks
  - DO, FOR, REPEAT
  - FUNCTION, PROCEDURE, METHOD

The code example below illustrates annotating the matching bracket.

```
IF DataObject <> '' THEN DO:
  setAppService('').
  startDataObject(DataObject). DataSourceHdl = 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 long 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.
Jumping to a matching element

With the cursor in or immediately to the right of a matched element, press **CTRL+SHIFT+P** to go to the 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

You can disable the bracket matching feature in the ABL Editor by deselecting the **Highlight matching elements in code** option in **OpenEdge Editor preferences**. Turning off this option does not disable jumping to matching elements with **CTRL+SHIFT+P**.

Related Concepts

- Code assistance

Related Tasks

- Setting Editor options
- Using code-completion features

Related Reference

- Editor Assistance options
Using code-completion features

Code-completion assistance gives you suggestions for completing the code that you are typing in the ABL Editor. You get code-completion assistance when you press **CTRL+SPACE**.

Code-completion assistance filters suggestions based on context. Code-completion assistance for keywords shows relevant keywords only. Repeatedly pressing **CTRL+SPACE** toggles between context-filtered suggestions and all suggestions.

In addition to ABL keywords, code completion assistance provides 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

In addition to invoking completion assistance at any time by pressing **CTRL+SPACE**, you can take advantage of an option to 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 **Auto-completion activation on '.' and ':''** option in Editor Assistance preferences.

Balancing code-completion convenience and performance

To propose a complete and accurate set of completion options, OpenEdge Architect 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.

OpenEdge Architect 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.

**Disabling database field descriptions to improve code-assistance performance**

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.

**Related Concepts**

- Code assistance

**Related Tasks**

- Using templates for easier text entry
- Displaying ABL syntax help

**Related Reference**

- Editor Assistance options
Using templates for easier text entry

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.

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.

To manage the set of available templates, go to the **Templates (Macros)** page in Editor preferences:

1. Choose **Window>Preferences**. The **Preferences** window appears.
2. Choose **OpenEdge Architect>Editor>Templates (Macros)**. The **Templates** page appears.
3. The page displays the list of existing templates. To modify a template, click **Edit** and make the desired changes. Click **Apply** to save your changes.

   At this page, you can also click **New** to create templates, click **Import** 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.

**Related Tasks**

Using code-completion features

**Related Reference**

Templates (macros)
Editor Templates (Macros) options
Writing ABL code

This section includes the following topics:

- Creating ABL source files
- Adding functions
- Adding internal procedures
- Checking syntax
- Commenting out code
- Creating new source files
- Adding annotations to source files
- Adding Sonic ESB annotations
- Including database schema elements
- Opening included files
- Removing AppBuilder markup from files
- Setting breakpoints while editing
- Writing object-oriented ABL code
Using structured error handling

Many OpenEdge Architect 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` menu. The blocks are also included in code-assistance completion proposals where applicable.

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.

**Related Concepts**
- Code assistance

**Related Tasks**
- Creating new class files
- Creating new ABL source files
- Adding functions
- Adding internal procedures
- Adding methods

**Related Reference**
- ABL Language Reference
Adding functions

The OpenEdge Editor provides a wizard that helps you add a new function to a ABL code file. To use this wizard:

1. Choose **Source>Add Function.** The **Add Function** dialog appears.
2. Enter a name for the function, 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.
3. 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.

4. Click **OK.**

The OpenEdge Editor inserts a prototype for the function. You can fill in and modify the prototype as appropriate.

**Related Tasks**

- Creating new ABL source files
- Adding internal procedures
- Using structured error handling

**Related Reference**

- Add Function wizard
- ABL Language Reference
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. Enter a **Procedure name**. Also choose whether to include a **CATCH** block and/or a **FINALLY** block for structured error handling.
3. 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.
   - **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.

4. Click **OK**.

The OpenEdge Editor inserts the internal procedure.

**Related Tasks**

- Adding functions
- Creating new ABL source files
- Using structured error handling

**Related Reference**

- ABL Language Reference
Checking syntax

You can check the syntax of the active file in the Editor by typing `CTRL+SHIFT+C` or choosing **Check Syntax** from the pop-up context menu.

**Related Tasks**
- Clearing OpenEdge errors
- Compiling ABL code
- Displaying ABL syntax help
- Running ABL code
Commenting out code

You can quickly comment out and uncomment blocks of your code:

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 Editor changes the text color to match the settings in the Editor Color preferences.

**Related Tasks**
- Using basic ABL Editor functions
- Automatically formatting code

**Related Reference**
- Editor Colors options
Creating new ABL source files

OpenEdge Architect 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)

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>Other**.
3. Select the desired file type under the **OpenEdge>**Editor** node 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.

**Related Tasks**

- Writing object-oriented code
- Adding functions
- Adding internal procedures
- Compiling ABL code
- Running ABL code

**Related Reference**

- Code-generation wizards
Adding annotations to source files

Annotations are metadata that you can add to your 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 OpenEdge Architect main menu bar.

   In a file that is open in the ABL editor, you can right-click and choose Source>Add Annotation or OpenEdge>Add Annotation from the context menu. You can also right-click in the Resources view and choose OpenEdge>Add Annotation.

2. Select an Annotation template from the combo-box.

   You can modify the annotation text of the selected annotation in the text area provided below the annotation. To create your own annotations, see Adding templates for code annotations.

3. Check the files to which you want to add the annotation in the Available Resources treeview. 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.

4. Click Finish to add top level annotations to the selected files. Click Next if you want to add annotations to specific procedures or functions.

   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.

Related Concepts

Code annotations

Related Tasks

Adding templates for code annotations
Adding Sonic ESB annotations

Related Reference

Add Annotation wizard
Editor Annotations options
ABL Annotation content builder
ABL annotation syntax
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.

Applying ESB annotations to ABL code files

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. Select a project and open a .p, .i, or .w file.
2. Choose Source>Add Annotation from the OpenEdge Architect main menu bar.
   - 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 Resources view and choose OpenEdge>Add Annotation. The Add Annotations wizard appears.
3. Choose ESB Annotation-Main, which is selected by default.

   ![Select annotation or enter annotation text in editor](image)

   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.
4. Click Edit if you want to change the ESB Annotation-Main definition.

   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.

5. In the **Available Resources** section, select the files where you want to add ESB Annotation-Main. For example:

6. To save a copy of files before annotations are added, select the backup option and specify the save location. For example:

7. 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.

8. Click **Next** if you intend to add ESB annotations to internal procedures or user-defined functions.

   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.

9. Choose **ESB Annotation-Detail**, which is the appropriate annotation for internal procedures and user-defined functions, and which is selected by default.

10. Click **Edit** if you want to change the ESB Annotation-Detail definition.

   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.

11. 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:

12. Click **Finish**.

The ESB Annotation-Detail annotation is added to the selected internal procedures and user-defined functions.

You can also add ESB annotations from the **Outline** view:

1. Right-click on an internal procedure or user-defined function in the **Outline** view. (The procedure or function cannot be private.)

The **Outline** view is a tree view of the contents of the ABL code file that currently has focus.

2. Select **New ESB Annotation** from the context menu.

The **New ESB Annotation** dialog appears. It allows you to change the **ESBOE file name**, **Use return value**, and **Write dataset before image** attributes for the main and detail annotations.

3. Click **OK**.

The ESB Annotation-Main is added to the top of the code file and the ESB Annotation-Detail is added to the selected procedure or user-defined function.

**Generating ESBOE files**

To generate ESBOE files:

1. Right-click on the project name in the **Resources** view.

If you do not want to generate ESBOE files for every file in the project, you can select individual file names in the **Resources** view.

You can also right-click in a file that is open in the ABL Editor.

2. Choose **OpenEdge>Generate Sonic ESB Invocation Files**.

**Related Concepts**

**Code annotations**

For more information about Sonic applications see OpenEdge Development: Messaging and ESB in the Product Documentation section of the Progress Software Developer's Network Web site.
**Related Tasks**

- Adding templates for code annotations

**Related Reference**

- Editor Annotations options
- ABL Annotation content builder
- ABL annotation syntax
Including database schema elements

The ABL Editor includes several features to make it easier to add database schema elements, such as tables and fields. These features include:

- Expanding table and field names
- Automatically prefixing the database name
- Dragging and dropping table names from the DB Structure view
- Copying and pasting other schema element names from the DB Structure view

**Related Tasks**

Using code-completion features

**Related Reference**

Editor Assistance options
Opening included files

There are several methods for opening included files in the OpenEdge Editor:

- Click on the include file in the **Outline** view.
- Double-click the include file preprocessor in the code.
- Select the include file in the **Show References** dialog and click **Open**.

To open the **Show References** dialog, right-click in the source file to open the Editor's pop-up context menu and choose **Show References**.
Removing AppBuilder markup from files

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 Editor’s context menu.

Related Tasks
Automatically formatting code
Commenting out code
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 ☰ appears in the left margin, indicating a breakpoint. If the line was previously marked, the breakpoint is removed, and the blue circle disappears.

**Related Concepts**
- Breakpoints

**Related Tasks**
- Using breakpoints
Writing object-oriented ABL code

*Demo*

Navigating Classes in Architect, Part 1: Using the ABL Editor

This section includes the following topics:

- Creating new class files
- Creating new interface files
- Adding constructors
- Adding constructors from a super class
- Adding a destructor
- Adding methods
- Overriding methods
- Coding methods implemented via interface
- Adding properties
Creating new class files

Use the following procedure to create a new class file:

1. Choose **File>New>ABL Class** to launch the New ABL Class wizard. Tooltips appear in the title pane to describe the purpose of the field that has focus.
2. Fill in the appropriate information for the new class.
3. Click **Finish**. The ABL Editor opens the new class file.

**Note:** The class cannot inherit from a generic super class, nor can it implement a generic interface.

**Related Concepts**

- ABL object-oriented extensions

**Related Tasks**

- Using structured error handling
- Creating new interface files
- Creating new ABL source files

**Related Reference**

- New ABL Class wizard
Creating new interface files

Use the following procedure to create a new interface file:

1. Choose **File>New>ABL Interface** to launch the New ABL Interface wizard. Tooltips appear in the title pane to describe the purpose of the field that has focus.
2. Fill in the appropriate information for the new interface.
3. Click **Finish**. The ABL Editor opens the new interface file.

**Related Concepts**
- ABL object-oriented extensions

**Related Tasks**
- Coding members implemented via interface or abstraction
- Overriding members
- Creating new class files

**Related Reference**
- New ABL Interface wizard
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.

**Related Concepts**

ABL object-oriented extensions

**Related Tasks**

Using structured error handling
Adding constructors from a super class
Adding a destructor

**Related Reference**

Add Constructor wizard
ABL Language Reference
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, use the following procedure:

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.

**Related Concepts**
- ABL object-oriented extensions

**Related Tasks**
- Adding constructors
- Adding a destructor

**Related Reference**
- Add Constructors from Super Class wizard
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.

**Related Concepts**

- ABL object-oriented extensions

**Related Tasks**

- Adding constructors
- Adding constructors from a super class

**Related Reference**

- ABL Language Reference
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**.

OpenEdge Architect 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 method** - Insert the code prototype immediately before/after the first/last method currently in the class.
- **After name** - Insert the code prototype after the named method.

**Related Concepts**

ABL object-oriented extensions

**Related Tasks**

Overriding methods
Coding methods implemented via interface
Adding properties

**Related Reference**

Add Method wizard
ABL Language Reference
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**.

OpenEdge Architect inserts the event at the place specified in the **Insertion position** field. You have the following options:

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

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

- **Cursor position** - Insert the code prototype at the current position of the cursor in the file.
- **First/Last event** - Insert the code prototype immediately before/after the first/last event currently in the class.
- **After name** - Insert the code prototype after the named event.

**Related Concepts**

ABL object-oriented extensions

**Related Reference**

Add Event wizard
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 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 property.

**Related Concepts**

ABL object-oriented extensions

**Related Tasks**

- Adding methods
- Adding constructors
- Adding a destructor

**Related Reference**

Add Property wizard
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.

**Related Concepts**

ABL object-oriented extensions

**Related Tasks**

- Adding a destructor
- Adding constructors
- Adding constructors from a super class
- Adding methods
- Adding properties
- Coding methods implemented via interface

**Related Reference**

Override/Implement Members wizard
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**.
2. 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.
3. Select either **Generate exceptions for required members** or **Generate default values for required members**.
4. Click **Generate** to add the selected member stubs to the class file.

**Related Tasks**

- Creating new class files
- Creating new interface files
- Overriding methods

**Related Reference**

- New ABL Class wizard
Override/Implement Members wizard
Reference

This section includes the following topics:

Editor Menus
ABL Editor preferences
Hot keys
Templates (macros)
ABL Editor preferences

This section includes the following topics:

- Editor options
- Editor Annotations options
- Editor Assistance options
- Editor Build options
- Editor Colors options
- Editor Templates (Macros) options
- Class Cache options
# Editor options

The **Editor** page (**OpenEdge Architect>Editor**) of the Editor preferences includes the following options:

<table>
<thead>
<tr>
<th>Configure tab-space preference</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>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:</td>
<td></td>
</tr>
<tr>
<td>• <strong>Displayed tab width</strong> sets the number of spaces between tab stops. Note that in all cases, pressing <strong>TAB</strong> 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.</td>
<td></td>
</tr>
<tr>
<td>• <strong>Insert spaces for tabs</strong> determines whether white space inserted before a tab stop consists of individual space characters or a single tab character.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Replace content when tab is pressed in overwrite mode</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>When <strong>Insert spaces for tabs</strong> is selected in Eclipse Text Editor preferences, determines whether pressing <strong>TAB</strong> 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 <strong>Insert spaces for tabs</strong> is not selected, pressing <strong>TAB</strong> always inserts tab spacing, whether in insert mode or in overwrite mode.)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Case radio set</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies what keyword casing the ABL Editor applies: when you use the <strong>Correct Case</strong> command; as you type (if <strong>Expand keywords</strong> is enabled), or when you save a file (if <strong>Apply keyword casing on save</strong> is enabled).</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expand keywords</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determines whether the ABL Editor expands an ABL keyword when you type one of the abbreviations that the AVM accepts. For example, when selected, the Editor expands &quot;DEF&quot; to &quot;DEFINE&quot; as soon as you type a space.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Case keywords</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determines whether the ABL Editor immediately applies keyword casing when you type a space following a full or abbreviated keyword.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Apply keyword casing on save</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determines whether the ABL Editor applies keyword casing when you save the file.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Expand database tables and fields</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<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>
<td></td>
</tr>
</tbody>
</table>
**Automatic prefix fields and tables with database name**
Determines whether the ABL Editor qualifies database tables and fields with the database alias. **Note:** If multiple aliases are defined for a database connection, the first from the list is used.

**Enable smart home and end keys**
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.

**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.

**Apply** stores your changes to the current workspace configuration. **Restore Defaults** resets the preferences to the OpenEdge Architect defaults.

**Related Tasks**
- Setting ABL Editor preferences

**Related Reference**
- Editor Annotations options
- Editor Assistance options
- Editor Build options
- Editor Colors options
- Editor Templates (Macros) options
- Class Cache options
Editor Annotations options

The Annotations page (OpenEdge Architect>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 Add Annotation wizard. To browse through long lists, select a template and then drag the cursor up or down.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotation text</td>
<td>Displays the text of the selected template. A template can include more than one annotation. <strong>Note:</strong> 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.</td>
</tr>
<tr>
<td>Add</td>
<td>Launches the New Annotation dialog. <strong>Note:</strong> The New Annotation dialog does not check the syntax of the template you create.</td>
</tr>
<tr>
<td>Edit</td>
<td>Launches the Edit Annotation dialog for the selected annotation. <strong>Note:</strong> The Edit Annotation dialog does not check the syntax of the template you create.</td>
</tr>
<tr>
<td>Remove</td>
<td>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.</td>
</tr>
<tr>
<td>Import</td>
<td>Overwrites the current annotation templates with the contents of an external .dat file.</td>
</tr>
<tr>
<td>Export</td>
<td>Saves the current annotation templates to an external .dat file.</td>
</tr>
</tbody>
</table>

**Apply** stores your changes to the current workspace configuration. **Restore Defaults** resets the preferences to the OpenEdge Architect defaults.

**Related Tasks**

Adding annotations to source files

**Related Reference**

Introducing the Meta Catalog
Add Annotation wizard
## Editor Assistance options

The **Assistance** page (OpenEdge Architect>Editor>Assistance) of the Editor preferences includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Show syntax help as I type</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Highlight matching elements in code</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Show description when proposing schema elements</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>For schema</strong></td>
<td>If checked, enables display of a pop-up box identifying schema elements when you leave the cursor over them.</td>
</tr>
<tr>
<td><strong>For keyword</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Milliseconds allowed for re-parsing</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Automatically propose completion on &quot;.&quot; and &quot;:&quot;</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Auto-bracketing</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Exclude proposals (list)</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Order of proposals (list)</strong></td>
<td>Specifies the order in which assistance proposals are presented. To change the sequence, select an element and click <strong>Move Up</strong> or <strong>Move Down</strong>.</td>
</tr>
</tbody>
</table>

**Apply** stores your changes to the current workspace configuration. **Restore Defaults** resets the preferences to the OpenEdge Architect defaults.
**Related Concepts**

- Code parsing
- Code assistance

**Related Tasks**

- Using code-completion features
- Displaying ABL syntax help
- Setting Eclipse editor preferences
- Matching paired elements

**Related Reference**

- ABL Editor preferences
- Hot keys
- Templates (macros)
**Editor Build options**

The **Build** page (*OpenEdge Architect > 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,dbconnections</code>.</td>
</tr>
<tr>
<td><strong>Compile on save if required</strong></td>
<td>An option to have OpenEdge Architect compile ABL source files each time you save them. It is recommended that you select this option if you disable the <strong>Build automatically</strong> option in Eclipse Workspace preferences (<em>Window &gt; Preferences &gt; General &gt; Workspace</em>). If automatic builds are enabled (the default setting), the <strong>Compile on save if required</strong> option has no effect. 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 OpenEdge Architect 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>
<tr>
<td><strong>Show help message detail with syntax check messages</strong></td>
<td>An option to automatically include help text with each message listed in the Check Syntax dialog. Enabling this option has no effect unless <strong>Show all syntax check message summaries</strong> is also enabled.</td>
</tr>
</tbody>
</table>
### Related Tasks

- Setting Eclipse editor preferences
- Setting ABL Editor preferences

### Related Reference

- Editor options
- Editor Annotations options
- Editor Assistance options
- Editor Colors options
- Editor Templates (Macros) options
- Class Cache options
Editor Colors options

The Colors page (OpenEdge Architect>Editor>Colors) of the Editor preferences includes the following options:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Default setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>The color for all text not covered by another category. Select the <strong>Bold</strong> toggle to make all text in this category bold.</td>
<td>Black (0,0,0)</td>
</tr>
<tr>
<td>Keyword</td>
<td>The color for ABL keywords. Select the <strong>Bold</strong> toggle to make all text in this category bold.</td>
<td>Purple (127,0,85)</td>
</tr>
<tr>
<td>Schema</td>
<td>The color for schema elements, such as database tables and fields. Select the <strong>Bold</strong> toggle to make all text in this category bold.</td>
<td>Orange (255,128,0)</td>
</tr>
<tr>
<td>DataType</td>
<td>The color for datatypes. Select the <strong>Bold</strong> toggle to make all text in this category bold.</td>
<td>Red (205,58,58)</td>
</tr>
<tr>
<td>Preprocessor</td>
<td>The color for preprocessors. Select the <strong>Bold</strong> toggle to make all text in this category bold.</td>
<td>Brown (139,126,102)</td>
</tr>
<tr>
<td>Comment</td>
<td>The color for commented text. Select the <strong>Bold</strong> toggle to make all text in this category bold.</td>
<td>Green (63,127,95)</td>
</tr>
<tr>
<td>Include</td>
<td>The color for names of included files. Select the <strong>Bold</strong> toggle to make all text in this category bold.</td>
<td>Pink (255,0,128)</td>
</tr>
<tr>
<td>Literal</td>
<td>The color for hard-coded values, such as numbers and strings. Select the <strong>Bold</strong> toggle to make all text in this category bold.</td>
<td>Blue (42,0,255)</td>
</tr>
</tbody>
</table>

To change a default color choice for a specific element, click on the color button to launch the Color dialog. Select from the list of available basic and custom colors. To define a custom color, click **Define Custom Colors** to display the color palette.

**Apply** stores your changes to the current workspace configuration. **Restore Defaults** resets the preferences to the OpenEdge Architect defaults.

**Related Tasks**
- Setting Eclipse editor preferences
- Setting ABL Editor preferences
Related Reference

Editor options
Editor Annotations options
Editor Assistance options
Editor Build options
Editor Templates (Macros) options
Class Cache options
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>OpenEdge Architect>Editor>Templates (Macros).

This page provides the following options:

<table>
<thead>
<tr>
<th>Template list</th>
<th>Displays the available templates. Templates that are checked are available for use; those that are not checked are disabled.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preview</td>
<td>Displays the pattern for the selected template.</td>
</tr>
<tr>
<td>New</td>
<td>Launches the <strong>New Template</strong> dialog to add a template.</td>
</tr>
<tr>
<td>Edit</td>
<td>Launches the <strong>Edit Template</strong> dialog to enable you to edit the definition of the selected template.</td>
</tr>
<tr>
<td>Remove</td>
<td>Deletes the selected templates.</td>
</tr>
<tr>
<td>Restore Removed</td>
<td>Undoes any deletions.</td>
</tr>
<tr>
<td>Revert to Default</td>
<td>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 OpenEdge Architect.</td>
</tr>
<tr>
<td>Import</td>
<td>Imports an XML file of template definitions.</td>
</tr>
<tr>
<td>Export</td>
<td>Writes the selected template definitions to an XML file, abbreviations.xml by default.</td>
</tr>
<tr>
<td>Restore Defaults</td>
<td>Resets the list of templates to its as-installed state, undoing all modifications.</td>
</tr>
<tr>
<td>Apply</td>
<td>Stores your changes to the current workspace configuration.</td>
</tr>
</tbody>
</table>

**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.

**Related Tasks**
- Setting Eclipse editor preferences
- Setting ABL Editor preferences

**Related Reference**
- Editor options
- Editor Annotations options
- Editor Assistance options
- Editor Build options
Editor Colors options
Templates (Macros)
Class Cache options

The Class Cache page (OpenEdge Architect>Advanced>Class Cache) of the OpenEdge Architect preferences includes the following options:

<table>
<thead>
<tr>
<th>Limit scope of cache class information to</th>
<th>Defines what directories are searched for classes and interfaces when building the cache. With the Workspace option, OpenEdge Architect 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 OpenEdge Architect slower, as building the class cache takes longer.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scan installation directory at startup for procedure information</td>
<td>If selected, tells OpenEdge 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.</td>
</tr>
</tbody>
</table>

Apply stores your changes to the current workspace configuration. Restore Defaults resets the preferences to the OpenEdge Architect defaults.

Related Tasks

Setting ABL Editor preferences

Related Reference

Editor options
Editor Annotations options
Editor Assistance options
Editor Build options
Editor Colors options
Editor Templates (Macros) options
PROPATH and source settings
ABL Editor menus

This section includes the following topics:

Source menu
Editor context menu
## Source menu

The **Source** menu contains options specific to the ABL Editor. The **Source** menu is available from the Workbench menu bar and from the editor's right-click context menu.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check Syntax</td>
<td>Check the ABL syntax of the current file. This option does not save an r-code version of the file.</td>
</tr>
<tr>
<td>Compile</td>
<td>Build the current file and save the r-code.</td>
</tr>
<tr>
<td>Toggle Comment</td>
<td>Comment or uncomment the selected lines of code.</td>
</tr>
<tr>
<td>Correct Case</td>
<td>Apply the user preference for keyword casing to keywords in the current file.</td>
</tr>
<tr>
<td>Expand Keywords</td>
<td>Replace abbreviated keywords in the current file with their fully spelled equivalents.</td>
</tr>
<tr>
<td>Correct Indentation</td>
<td>Indent lines in the current file to improve readability.</td>
</tr>
<tr>
<td>Update Function Prototypes</td>
<td>Modify 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>Remove the AppBuilder markup code from a file originally created in AppBuilder or by means of the New ABL Structured Include or New ABL Structured Procedure wizard. <strong>Caution:</strong> If this code is removed, the file cannot be opened in AppBuilder again.</td>
</tr>
<tr>
<td>Add Constructor</td>
<td>Open the <strong>Add Constructor wizard</strong> to add a constructor to the current class.</td>
</tr>
<tr>
<td>Add Static Constructor</td>
<td>Add a static constructor to the current class.</td>
</tr>
<tr>
<td>Add Constructors from Super Class</td>
<td>Open the <strong>Add Constructors from Super Class</strong> 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>Add a destructor to the current class.</td>
</tr>
<tr>
<td>Add Method</td>
<td>Open the <strong>Add Method wizard</strong> to add a method to the current class.</td>
</tr>
<tr>
<td>Add Event</td>
<td>Open the <strong>Add Event wizard</strong> to add an event to the current class.</td>
</tr>
<tr>
<td>Add Property</td>
<td>Open the <strong>Add Property wizard</strong> to add a property to the current class.</td>
</tr>
<tr>
<td>Override/Implement Members</td>
<td>Open the <strong>Override/Implement Members wizard</strong> to select members from the class hierarchy to override or implement in the current class.</td>
</tr>
<tr>
<td><strong>Surround With</strong></td>
<td>Show a submenu that lets you surround the selected code segment with a <strong>CATCH</strong> block or a <strong>FINALLY</strong> block for structured error handling.</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Add Procedure</strong></td>
<td>Open the Add Procedure wizard to add an internal procedure to the current procedure.</td>
</tr>
<tr>
<td><strong>Add Function</strong></td>
<td>Open the Add Function wizard to add a function to the current procedure.</td>
</tr>
<tr>
<td><strong>Add Annotation</strong></td>
<td>Open the Add Annotation wizard to insert ABL annotations in one or more files.</td>
</tr>
</tbody>
</table>

*Related Reference*

*Editor context menu*
Editor context menu

Right-clicking in a file displays the OpenEdge Editor's context menu.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Undo</strong></td>
<td>Undo the last action in the current file.</td>
</tr>
<tr>
<td><strong>Revert file</strong></td>
<td>Replace the current file with the local saved copy.</td>
</tr>
<tr>
<td><strong>Save</strong></td>
<td>Save the current file.</td>
</tr>
<tr>
<td><strong>View Design</strong></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><strong>Show In</strong></td>
<td>See the <em>Workbench User Guide</em>.</td>
</tr>
<tr>
<td><strong>Quick Outline</strong></td>
<td>Open the <em>Quick Outline</em> window.</td>
</tr>
<tr>
<td><strong>Cut</strong></td>
<td>Cut the selected text.</td>
</tr>
<tr>
<td><strong>Copy</strong></td>
<td>Copy the selected text.</td>
</tr>
<tr>
<td><strong>Paste</strong></td>
<td>Paste the contents of the Clipboard.</td>
</tr>
<tr>
<td><strong>Shift Right</strong></td>
<td>Increase the indent of the current line or the highlighted lines.</td>
</tr>
<tr>
<td><strong>Shift Left</strong></td>
<td>Decrease the indent of the current line or the highlighted lines.</td>
</tr>
<tr>
<td><strong>Source</strong></td>
<td>Display the OpenEdge Architect <em>Source menu</em>.</td>
</tr>
<tr>
<td><strong>Check Syntax</strong></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><strong>Compile</strong></td>
<td>Compile the current file and save the r-code.</td>
</tr>
<tr>
<td><strong>Keyword Help</strong></td>
<td>Open the <em>Keyword Help</em> view in the <em>Console</em> area, displaying the <em>OpenEdge Development: ABL Reference</em> entry for the selected keyword.</td>
</tr>
<tr>
<td><strong>Toggle Breakpoint</strong></td>
<td>Set or remove a breakpoint at the current line for the OpenEdge Debugger.</td>
</tr>
<tr>
<td><strong>Show References</strong></td>
<td>Open the <em>References</em> dialog which displays the external references in the current file.</td>
</tr>
<tr>
<td><strong>Open Declaration</strong></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><strong>Run As</strong></td>
<td>Display the OpenEdge Architect <em>Run</em> menu.</td>
</tr>
<tr>
<td><strong>Debug As</strong></td>
<td>Display a submenu of appropriate debuggers for the current file.</td>
</tr>
<tr>
<td>Team</td>
<td>See the <em>Workbench User Guide</em>.</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>Compare With</td>
<td>See the <em>Workbench User Guide</em>.</td>
</tr>
<tr>
<td>Replace With</td>
<td>See the <em>Workbench User Guide</em>.</td>
</tr>
<tr>
<td>OpenEdge</td>
<td>Display a submenu of common OpenEdge actions.</td>
</tr>
<tr>
<td>Preferences</td>
<td>Launch the <em>Preferences</em> window displaying the <em>General&gt;Editors&gt;Text Editors</em> preference page. See the <em>Workbench User Guide</em> for more information on these settings.</td>
</tr>
</tbody>
</table>

**Related Reference**

*Source menu*
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. It includes the following topics:

New ABL Class wizard
New ABL Include wizard
New ABL Interface wizard
New ABL Procedure wizard
New ABL Structured Include wizard
New ABL Structured Procedure wizard
Add Constructor wizard
Add Constructors from Super Class wizard
Add Method wizard
Add Event wizard
Add Property wizard
Override/Implement Members wizard
Add Procedure wizard
Add Function wizard
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><strong>Package root</strong></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><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>Class 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 <strong>FINAL</strong> 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>
<tr>
<td><strong>Inherits</strong></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. <strong>Note:</strong> The super class cannot be generic.</td>
</tr>
<tr>
<td><strong>Implements</strong></td>
<td>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. <strong>Note:</strong> The class cannot implement a generic interface.</td>
</tr>
<tr>
<td><strong>Generate default constructor</strong></td>
<td>If checked, specifies that the class is to include a default constructor method.</td>
</tr>
</tbody>
</table>
Generate destructor | If checked, specifies that the class is to include a destructor method.

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.

Related Concepts

ABL object-oriented extensions

Related Tasks

Writing object-oriented ABL code
Developing a visual container
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:

<table>
<thead>
<tr>
<th><strong>Container</strong></th>
<th>Specifies a currently open project or a folder within the project where the include 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><strong>File name</strong></td>
<td>(Required) Specifies the name of the include file. The <code>.i</code> extension is appended automatically.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Lets you optionally enter a description for the file. This text will appear in the file header.</td>
</tr>
<tr>
<td><strong>Purpose</strong></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><strong>Author</strong></td>
<td>Lets you optionally enter the author's name. This text will appear in the file header.</td>
</tr>
</tbody>
</table>

**Related Tasks**

- Creating ABL source files
- Writing object-oriented ABL code
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>Package root</th>
<th>Specifies a currently open project to contain the interface code and other project code. Click Browse 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 interface file. Click Browse 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>Interface name</td>
<td>(Required) Specifies 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>Description</td>
<td>Lets you optionally enter a description for the interface. 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 interface. This text will appear in the file header.</td>
</tr>
</tbody>
</table>

**Related Concepts**

ABL object-oriented extensions

**Related Tasks**

Creating ABL source files
Writing object-oriented ABL code
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>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 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>
</tbody>
</table>

**Related Concepts**

ABL object-oriented extensions

**Related Tasks**

Creating ABL source files
Writing object-oriented ABL code
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>ABL Structured Include**.

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 include 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 .i 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>

**Related Tasks**

- Creating new ABL source files
- OpenEdge AppBuilder
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>ABL Structured Procedure**.

The following controls are available:

<table>
<thead>
<tr>
<th><strong>Container</strong></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><strong>File name</strong></td>
<td>(Required) Specifies the name of the include file. The .p extension is appended automatically.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Lets you optionally enter a description for the file. This text will appear in the file header.</td>
</tr>
<tr>
<td><strong>Purpose</strong></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><strong>Author</strong></td>
<td>Lets you optionally enter the author's name. This text will appear in the file header.</td>
</tr>
</tbody>
</table>

**Related Tasks**

- Creating ABL source files
- OpenEdge AppBuilder
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><strong>Public/Protected/Private</strong></th>
<th>Specifies the access modifier that applies to the method:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• <strong>Public</strong> — The constructor can be called from any class and directly invoked in any derived class.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Protected</strong> — The constructor can be directly invoked in any derived class, but it cannot be called from a class that is not derived from the declaring class.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Private</strong> — The constructor cannot be called from any other class and cannot be directly invoked in a derived class.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Insert catch block</strong></th>
<th>If checked, adds a CATCH block to the generated method stub to support structured error handling.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Insert finally block</strong></td>
<td>If checked, adds a FINALLY block to the generated method stub to support structured error handling.</td>
</tr>
<tr>
<td><strong>Insertion position</strong></td>
<td>Determines where the constructor code is inserted in the source file.</td>
</tr>
</tbody>
</table>

**Related Concepts**

ABL object-oriented extensions

**Related Tasks**

Adding constructors
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><strong>Constructor list</strong></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><strong>Insertion position</strong></td>
<td>Determines where the constructor code is inserted in the source file.</td>
</tr>
<tr>
<td><strong>Generate method comments</strong></td>
<td>If checked, adds a comment block before the inserted method code.</td>
</tr>
</tbody>
</table>

**Related Concepts**

- ABL object-oriented extensions

**Related Tasks**

- Adding constructors from a super class
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>
<tbody>
<tr>
<td>Public/Protected/Private</td>
<td>Specifies the access modifier that applies to the method:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Public</strong> — The method can be called from any class and can be directly invoked in any derived class.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Protected</strong> — 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.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Private</strong> — The method cannot be called from any other class and cannot be directly invoked in a derived class. A private method cannot be abstract.</td>
</tr>
<tr>
<td>Abstract</td>
<td>(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.</td>
</tr>
<tr>
<td>Static</td>
<td>If checked, specifies that the method 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 method cannot be abstract.</td>
</tr>
<tr>
<td>Return type</td>
<td>Specifies the method's data type.</td>
</tr>
<tr>
<td>Insert catch block</td>
<td>If checked, adds a CATCH block to the generated method stub to support structured error handling.</td>
</tr>
<tr>
<td>Insert finally block</td>
<td>If checked, adds a FINALLY block to the generated method stub to support structured error handling.</td>
</tr>
<tr>
<td><strong>Insertion position</strong></td>
<td>Determines where the method code is inserted in the source file.</td>
</tr>
</tbody>
</table>

**Related Concepts**
ABL object-oriented extensions

**Related Tasks**
Adding methods
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:

<table>
<thead>
<tr>
<th><strong>Event name</strong></th>
<th>(Required) Specifies the name of the event. 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 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. |

**Related Concepts**

- ABL object-oriented extensions

**Related Tasks**

- Adding events
## 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>Property name</th>
<th>(Required) Specifies the name of the property. 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 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. |
<p>| <strong>Abstract</strong> | (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. |
| <strong>Type</strong> | Specifies the property's data type. |
| <strong>Extent</strong> | 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. |
| <strong>Initial value</strong> | If checked, enables the adjacent field to the right where you enter the initial value of the property. |
| <strong>NO-UNDO</strong> | If checked, adds the NO-UNDO attribute to the property declaration. |</p>
<table>
<thead>
<tr>
<th><strong>Readable</strong></th>
<th>If checked, specifies that the GET accessor is readable.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Writable</strong></td>
<td>If checked, specifies that the SET accessor is writable.</td>
</tr>
<tr>
<td><strong>Insert implementation</strong></td>
<td>If checked, adds an implementation stub for the accessor to the generated code.</td>
</tr>
<tr>
<td><strong>Insertion position</strong></td>
<td>Determines where the property code is inserted in the source file.</td>
</tr>
</tbody>
</table>

**Related Concepts**

ABL object-oriented extensions

**Related Tasks**

Adding properties
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>Tree view</th>
<th>Displays and lets you select data members in the class hierarchy that are eligible to be overridden or implemented.</th>
</tr>
</thead>
<tbody>
<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 &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 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>

Related Concepts

ABL object-oriented extensions

Related Tasks

Coding members implemented via interface or abstraction
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:

<table>
<thead>
<tr>
<th><strong>Procedure name</strong></th>
<th>(Required) Specifies the name of the procedure.</th>
</tr>
</thead>
<tbody>
<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>Insertion position</strong></td>
<td>Determines where the procedure code is inserted in the source file.</td>
</tr>
</tbody>
</table>

**Related Tasks**

Adding internal procedures
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:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>(Required) Specifies the name of the function.</td>
</tr>
<tr>
<td>Insert catch block</td>
<td>If checked, adds a CATCH block to the generated procedure stub to support structured error handling.</td>
</tr>
<tr>
<td>Return type</td>
<td>Specifies the data type of the value returned by the function.</td>
</tr>
<tr>
<td>Extent</td>
<td>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.</td>
</tr>
<tr>
<td>Insert catch block</td>
<td>If checked, adds a CATCH block to the generated function stub to support structured error handling.</td>
</tr>
<tr>
<td>Insert finally block</td>
<td>If checked, adds a FINALLY block to the generated function stub to support structured error handling.</td>
</tr>
<tr>
<td>Insertion position</td>
<td>Determines where the function code is inserted in the source file.</td>
</tr>
</tbody>
</table>

Related Tasks

Adding functions
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 **OpenEdge>Add Annotation** from the context menu for the Resources view or the ABL Editor. The following controls are available:

<table>
<thead>
<tr>
<th>Annotation selector</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotation template editor</td>
<td>Displays and lets you edit the content of the annotation to be inserted. The <strong>Edit</strong> button opens a dialog that supports changes specific to ESB annotations; this button is not active for other annotation types.</td>
</tr>
<tr>
<td>Available resources filter</td>
<td>Lets you filter the list of resources displayed in the treeview below.</td>
</tr>
</tbody>
</table>
| Available resources list | **On the first page of the wizard** - Lets 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** - Lets 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. |

**Related Tasks**
- Adding annotations to source files
- Adding Sonic ESB annotations

**Related Reference**
- Editor Annotations options
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>Show references</td>
<td>CTRL+SHIFT+G</td>
</tr>
<tr>
<td>Surround With Catch Block</td>
<td>ALT+SHIFT+Z, C</td>
</tr>
<tr>
<td>Surround With Finally Block</td>
<td>ALT+SHIFT+Z, F</td>
</tr>
<tr>
<td>Toggle Comment</td>
<td>CTRL+/</td>
</tr>
</tbody>
</table>
Related Tasks

- Writing ABL code
- Using basic ABL Editor functions
OpenEdge Editor perspective views

The default OpenEdge Editor perspective contains several views. Some are standard Workbench views. Some are specific to the OpenEdge Architect. 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 Workbench text editor customized for working with ABL. Its behavior is controlled by both general Workbench preferences and OpenEdge-specific preferences.
- **Resources** - A hierarchical view of the projects and resources in a workspace. It 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. It 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 open in the OpenEdge Editor.
- **Console** - A Workbench view that displays the text output from commands (runtime 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. It is the standard Workbench Problems view.
- **Tasks** - A to-do list. It is the standard Workbench Tasks view.

*Note:* The links in the above list take you to the *Workbench User Guide* for more information on standard Eclipse features.
Templates (macros)

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.

Note: Templates were formerly called macros in OpenEdge Architect, 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.

You can customize the available templates through the Templates (Macros) preferences page.

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><code>{&amp;FRAME-NAME} &amp;{cursor}</code></td>
</tr>
<tr>
<td>&amp;GL</td>
<td>Global name</td>
<td><code>{GLOBAL-DEFINE} &amp;{cursor}</code></td>
</tr>
<tr>
<td>&amp;SC</td>
<td>Scope defined</td>
<td><code>{SCOPED-DEFINE} &amp;{cursor}</code></td>
</tr>
<tr>
<td>CMT</td>
<td>Comment</td>
<td><code>/* &amp;{cursor} */</code></td>
</tr>
<tr>
<td>DED</td>
<td>Event with delegate</td>
<td><code>DEFINE EVENT &amp;{cursor} DELEGATE</code></td>
</tr>
<tr>
<td>DES</td>
<td>Event with signature</td>
<td><code>DEFINE EVENT &amp;{cursor} SIGNATURE</code></td>
</tr>
<tr>
<td>DVCH</td>
<td>Character variable</td>
<td><code>DEFINE VARIABLE &amp;{cursor} AS CHARACTER</code></td>
</tr>
<tr>
<td>DVDE</td>
<td>Decimal variable</td>
<td><code>DEFINE VARIABLE &amp;{cursor} AS DECIMAL</code></td>
</tr>
<tr>
<td>DVDT</td>
<td>Date variable</td>
<td><code>DEFINE VARIABLE &amp;{cursor} AS DATE</code></td>
</tr>
<tr>
<td>DVHN</td>
<td>Handle variable</td>
<td><code>DEFINE VARIABLE &amp;{cursor} AS HANDLE</code></td>
</tr>
<tr>
<td>DVIN</td>
<td>Integer variable</td>
<td><code>DEFINE VARIABLE &amp;{cursor} AS INTEGER</code></td>
</tr>
<tr>
<td>DVLG</td>
<td>Logical variable</td>
<td><code>DEFINE VARIABLE &amp;{cursor} AS LOGICAL</code></td>
</tr>
<tr>
<td>DVMP</td>
<td>CHAR variable</td>
<td><code>DEFINE VARIABLE &amp;{cursor} AS MEMPTR</code></td>
</tr>
<tr>
<td>DVRI</td>
<td>ROWID variable</td>
<td><code>DEFINE VARIABLE &amp;{cursor} AS ROWID</code></td>
</tr>
<tr>
<td>DVWH</td>
<td>Widget variable</td>
<td><code>DEFINE VARIABLE &amp;{cursor} AS WIDGET</code></td>
</tr>
<tr>
<td>ABL Editor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td><strong>IOPCH</strong></td>
<td>Character INPUT-OUTPUT</td>
<td>DEFINE INPUT-OUTPUT PARAMETER <code>${cursor}</code> AS CHARACTER</td>
</tr>
<tr>
<td><strong>IOPDE</strong></td>
<td>Decimal INPUT-OUTPUT</td>
<td>DEFINE INPUT-OUTPUT PARAMETER <code>${cursor}</code> AS DECIMAL</td>
</tr>
<tr>
<td><strong>IOPDT</strong></td>
<td>Date INPUT-OUTPUT</td>
<td>DEFINE INPUT-OUTPUT PARAMETER <code>${cursor}</code> AS DATE</td>
</tr>
<tr>
<td><strong>IOPHN</strong></td>
<td>Handle INPUT-OUTPUT</td>
<td>DEFINE INPUT-OUTPUT PARAMETER <code>${cursor}</code> AS HANDLE</td>
</tr>
<tr>
<td><strong>IOPIN</strong></td>
<td>Integer INPUT-OUTPUT</td>
<td>DEFINE INPUT-OUTPUT PARAMETER <code>${cursor}</code> AS INTEGER</td>
</tr>
<tr>
<td><strong>IOPLG</strong></td>
<td>Logical INPUT-OUTPUT</td>
<td>DEFINE INPUT-OUTPUT PARAMETER <code>${cursor}</code> AS LOGICAL</td>
</tr>
<tr>
<td><strong>IOPMP</strong></td>
<td>Memptr INPUT-OUTPUT</td>
<td>DEFINE INPUT-OUTPUT PARAMETER <code>${cursor}</code> AS MEMPTR</td>
</tr>
<tr>
<td><strong>IOPRI</strong></td>
<td>ROWID INPUT-OUTPUT</td>
<td>DEFINE INPUT-OUTPUT PARAMETER <code>${cursor}</code> AS ROWID</td>
</tr>
<tr>
<td><strong>IOPWH</strong></td>
<td>Widget INPUT-OUTPUT</td>
<td>DEFINE INPUT-OUTPUT PARAMETER <code>${cursor}</code> AS WIDGET</td>
</tr>
<tr>
<td><strong>IPCH</strong></td>
<td>Character INPUT</td>
<td>DEFINE INPUT PARAMETER <code>${cursor}</code> AS CHARACTER</td>
</tr>
<tr>
<td><strong>IPDE</strong></td>
<td>Decimal INPUT</td>
<td>DEFINE INPUT PARAMETER <code>${cursor}</code> AS DECIMAL</td>
</tr>
<tr>
<td><strong>IPDT</strong></td>
<td>Date INPUT</td>
<td>DEFINE INPUT PARAMETER <code>${cursor}</code> AS DATE</td>
</tr>
<tr>
<td><strong>IPHN</strong></td>
<td>Handle INPUT</td>
<td>DEFINE INPUT PARAMETER <code>${cursor}</code> AS HANDLE</td>
</tr>
<tr>
<td><strong>IPIN</strong></td>
<td>Integer INPUT</td>
<td>DEFINE INPUT PARAMETER <code>${cursor}</code> AS INTEGER</td>
</tr>
<tr>
<td><strong>IPLG</strong></td>
<td>Logical INPUT</td>
<td>DEFINE INPUT PARAMETER <code>${cursor}</code> AS LOGICAL</td>
</tr>
<tr>
<td><strong>IPMP</strong></td>
<td>Memptr INPUT</td>
<td>DEFINE INPUT PARAMETER <code>${cursor}</code> AS MEMPTR</td>
</tr>
<tr>
<td><strong>IPRI</strong></td>
<td>ROWID INPUT</td>
<td>DEFINE INPUT PARAMETER <code>${cursor}</code> AS ROWID</td>
</tr>
<tr>
<td><strong>IPWH</strong></td>
<td>Widget INPUT</td>
<td>DEFINE INPUT PARAMETER <code>${cursor}</code> AS WIDGET</td>
</tr>
<tr>
<td><strong>MES</strong></td>
<td>Message</td>
<td>MESSAGE <code>${cursor}</code></td>
</tr>
</tbody>
</table>
### Related Tasks

Using templates for easier text entry

### Related Reference

Editor Templates (Macros) options

<table>
<thead>
<tr>
<th>ABL</th>
<th>Description</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<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>
Introducing the OpenEdge Architect 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 OpenEdge Architect environment.

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. OpenEdge Architect 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

Creating a Form and a ProBindingSource
Adding a Grid Control to a Form
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
Concepts

This section includes the following topics:

Benefits of the Visual Designer
Code associated with a Visual Designer component
.NET assemblies and the Visual Designer
Visual container types
Implementing the OpenEdge GUI for .NET in your application
Application deployment
UI controls
The Visual Designer tool set
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 ProData-Sets).
- 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.

Related Concepts

- UI controls
- The Visual Designer tool set

Related Tasks

- Developing a visual container
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

OpenEdge Architect 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.

Related Tasks

Viewing and editing the container's source code
.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 OpenEdge Architect 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, OpenEdge Architect 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, OpenEdge Architect also adds a Referenced Assemblies node to the Resources 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 Resources view (or via the Project 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 Resources 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>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, OpenEdge Architect creates the assemblies.xml file in the project's root folder. To specify a different location, add -assemblies directory-path to the project startup parameters (Project>Properties>OpenEdge). You can enter the absolute directory path or the path relative to the project working directory.

All controls installed with OpenEdge Architect 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.

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 OpenEdge Architect by means of an option on the OpenEdge>Migration menu.

For more information, see Updating assembly references.

**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.

**Related Concepts**

UI controls

**Related Tasks**

Working with .NET resources
Adding and removing assembly references
Updating assembly references
Visual container types

As the starting point for building an OpenEdge GUI for .NET, you create a visual container by using one of three OpenEdge Architect wizards: New ABL Form, New ABL Dialog, or New ABL MDI Form.

**ABL Form**

The New ABL Form wizard creates a blank form.

![Visual Designer](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*.

**Related Tasks**
- Developing a visual container

**Related Reference**
- OpenEdge GUI for .NET wizards
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.

**Related Tasks**

- Developing a visual container
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.

**Related Concepts**

.NET assemblies and the Visual Designer

**Related Tasks**

Working with .NET resources
UI controls

All OpenEdge Architect users have access to a standard set of Microsoft .NET controls. These standard controls are included with the Microsoft .NET Framework, which the OpenEdge Architect 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.

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.

This section includes the following topics:

- Visual and non-visual controls
- Container controls
- Data-bound controls
- Custom user-defined controls
- Control properties
- Event behavior
- Access to controls
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 data grids.

- 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*.

**Related Concepts**

The Visual Designer tool set

**Related Tasks**

Working with UI controls
**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.

**Related Tasks**
- Developing a visual container
- Creating a user control
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.

Related Tasks

- Linking controls to data
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.

**Related Tasks**

- Creating a user control
- Creating an inherited control

**Related Reference**

- New ABL User Control wizard
- New ABL Inherited Control wizard
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.

Related Tasks

Defining properties and appearance

Related Reference

Visual Designer Properties view
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.

**Related Tasks**
- Coding event logic

**Related Reference**
- Visual Designer Properties view
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 OpenEdge Architect.

Related Concept
Visual Designer components

Related Tasks
Using the Toolbox
Sharing Toolbox settings

Related Reference
Visual Designer Editor
OpenEdge properties page
The Visual Designer tool set

This section describes the components of the Visual Designer and related tools. It includes the following topics:

- Visual Designer components
- The Visual Designer perspective
- Additional 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.

Global Toolbox

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 OpenEdge project 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, OpenEdge Architect 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.

**Related Tasks**
- Developing a visual container
- Working with UI controls
- Customizing the Visual Designer Toolbox
- Using the Outline view
- Setting project properties

**Related Reference**
- Workspace components
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.

Related Concepts
- Introducing the OpenEdge Class Browser

Related Tasks
- Developing a visual container

Related Reference
- Workspace components
Additional tools

OpenEdge Architect 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 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.

**Related Concepts**

Introducing the Class Browser  
Introducing the Meta Catalog  
Introducing the OpenEdge ABL Editor  
Using the Outline view  
OpenEdge Architect Debugger
Tasks

This section includes the following topics:

- Developing a visual container
- Customizing the Visual Designer Toolbox
- Working with .NET resources
Developing a visual container

This section includes the following topics:

- Creating the container
- Opening an existing container
- Sizing the container
- Viewing and editing the container's source code
- Deleting a container
- Working with UI controls
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**. OpenEdge Architect 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.

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: Advanced GUI Programming**.

**Related Concepts**

Visual container types

**Related Tasks**

Sizing the container
Working with UI controls
Creating custom controls

**Related Reference**

OpenEdge GUI for .NET wizards
Opening an existing container

To open an existing UI design class, right-click the file name in the Resources 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 Resources view. OpenEdge Architect 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.

**Related Tasks**

- Sizing the container
- Working with UI controls
Sizing the container

You can control the size of the form in either of two ways:

- 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.
- 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.

**Related Tasks**

Working with UI controls
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:

- Right-click in the Design Canvas and select **View Source** from the context menu.
- Select **Design>View Source**.
- 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:

- Right-click in the Design Canvas and select **View Design** from the context menu.
- Select **Design>View Design**.
- 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.

**Related Concepts**
- Code associated with a Visual Designer component

**Related Tasks**
- Coding event logic
Deleting a container

To delete a container class, select the .cls file in the Resources 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 ![show files](image), select **Filters**, and uncheck the **.resx** entry.

**Related Tasks**
- Deleting controls from a container
Working with UI controls

This section includes the following topics:

Using the Toolbox
Getting control-specific information
Using built-in design tools
Using the Undo and Redo options
Testing a UI design
Choosing controls for the UI
Defining properties and appearance
Linking controls to data
Coding event logic
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:

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: 🔄). 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.

Related Tasks

- Choosing controls for the UI
- Customizing the Visual Designer Toolbox

Related Reference

- Visual Designer Editor
- Custom user-defined controls
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 tool tip.

**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.
```
**Related Concepts**

Introducing the OpenEdge Class Browser

**Related Tasks**

Using the Properties view
Getting information from the Class Browser
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.

Related Tasks

Defining properties and appearance
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 Resources view and select **Replace With>Local History** from the context menu.

**Related Tasks**

Choosing controls for the UI
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.

**Related Tasks**

- Developing a visual container
- Running ABL code
Creating custom controls

This section includes the following topics:

- Creating a user control
- Creating an inherited control
Creating a user control

A user control is a composite control consisting of a container with individual controls that you add to it. You select the controls you need, define their properties and event logic, and save the container and its contents as a single entity that you can add to any form.

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. OpenEdge Architect 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.

**Related Concepts**
- Custom user-defined controls

**Related Tasks**
- Sizing the container
- Working with UI controls
- Defining properties and appearance
- Linking controls to data
- Coding event logic
- Customizing the Visual Designer Toolbox
**Related Reference**

New ABL User Control wizard
Creating an inherited control

<table>
<thead>
<tr>
<th>Demos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating a Data-bound Treeview as an Inherited Control (Part 1)</td>
</tr>
<tr>
<td>Creating a Data-bound Treeview as an Inherited Control (Part 2)</td>
</tr>
</tbody>
</table>

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**. OpenEdge Architect 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.

**Related Concepts**

- Custom user-defined controls

**Related Tasks**

- Defining properties and appearance
- Linking controls to data
- Coding event logic
- Customizing the Visual Designer Toolbox

**Related Reference**

- New ABL Inherited Control wizard
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, OpenEdge Architect 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.

Related Concepts

Custom user-defined controls

Related Tasks

Creating a user control
Creating an inherited control
Choosing controls for the UI

This section explains how to populate a visual container with controls. It includes the following topics:

- Adding controls to a container
- Deleting controls from a container
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 run time. 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.

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 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. 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 run time.

Related Tasks

- Deleting controls from a container
- Using the Undo and Redo options
- Using the Outline view
- Using the Toolbox
- Customizing the Visual Designer Toolbox
Related Reference

Visual Designer Editor
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 Resources 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** > **OpenEdge** > **Assemblies**). Directly editing the `assemblies.xml` file is not recommended.

**Note:** You cannot delete the visual container object (the form itself).

**Related Tasks**

- Adding controls to a container
- Selecting objects on the Design Canvas
- Using the Outline view
- Using the Undo and Redo options
- Deleting a container
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. It includes the following sections:

- Selecting objects on the Design Canvas
- Sizing objects
- Positioning objects on a form
- Setting tab order
- Using the Properties view
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.
Related Tasks

- Sizing objects
- Sizing the container
- Positioning objects on a form
- Setting tab order
- Using the Properties view
- Using the Outline view
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.

**Related Tasks**

- Selecting objects on the Design Canvas
- Positioning objects on a form
- Using the Undo and Redo options
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:

1. Select the base object, that is, the object whose dimension(s) you want to duplicate.
2. Keeping the base object selected, select the target objects to which you want to apply the dimension(s) of the base object.
3. **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.

### Related Tasks

- Selecting objects on the Design Canvas
- Using the Properties view
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

Directly moving objects

You can move a selected object or set of objects either by dragging with the mouse or by using the arrow keys.

The precise way the direct-moving techniques work depends on your Visual Designer preference settings.

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.

**Spacing and aligning objects**

The **Design** menu provides commands that facilitate regular spacing and alignment of controls.

**Spacing objects**

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.

**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:

1. Select the objects.
2. Select **Format>Center In Form** and choose the desired option.

To align objects to the grid:

1. Select the objects.
2. Select **Format>Align>To Grid**. The top left corner of each object snaps to the
nearest intersection of horizontal and vertical grid lines.

**Ordering 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 use these commands:

1. Select one or more objects on the form.
2. Select Design > Order and choose **Send to Back** or **Bring to Front**.

**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. 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 Diagram](image)

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.

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.

**Related Tasks**

- Selecting objects on the Design Canvas
- Using the Outline view
- Sizing objects
- Sizing the container
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.

   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.

**Related Tasks**

- Selecting objects on the Design Canvas
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.

**Viewing properties and events**

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.

**Sorting the list**

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**.

**Setting property values**

**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.

**Related Tasks**

- Selecting objects on the Design Canvas
- Coding event logic

**Related Reference**

- Visual Designer Properties view
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.

This section includes the following topics:

- Creating a binding source object
- Editing a ProBindingSource control
- Binding to an ABL data source
- Data binding example
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.

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.

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 Pro-DataSet or a temp table) from the Resources view to a form on the 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 desired schema elements 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 guidelines:

- Selecting a table also selects all of its columns.
- You can select multiple tables. However, all selected tables are placed at the top level when you drop them, and the finished ProBindingSource schema must have only one top-level(root) table. Therefore, you must edit the schema in the Designer before you can save it.
Visual Designer

- 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 table called Root Table.

- You cannot select a combination of tables and individual columns.

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 opens and displays the selected schema elements in the Tables and Fields panes.

Note: If the Resources 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 Resources 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. If necessary, edit the schema by using any of the techniques explained in the next section, starting with step 4.

5. Click OK to save the schema definition.

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 opens automatically.

3. Click Add Table. A top-level table node appears in the left 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 a schema, use the procedure explained below.

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 left or middle pane, and change the current values as appropriate by typing directly in the rightmost pane.

7. After making all desired changes, click OK to save the schema definition.

**Importing the 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 left 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.

4. At the Schema Selection dialog, select the specific schema elements that you want to import. Note the following restrictions:
   - You cannot select multiple tables.
   - You can select either an entire table or selected fields, but you cannot select both a table and a field.

5. Click the desired option (Append to current schema or Replace current schema).

6. Click OK. The ProBindingSource Designer opens with the selected elements in the left pane.

**Related Concepts**

- Data-bound controls

**Related Tasks**

- Editing a ProBindingSource control
- Binding to an ABL data source
- Data binding example
Related Reference

ProBindingSource Designer
Editing a ProBindingSource control

You can use the ProBindingSource Designer to edit the schema and schema properties of an existing binding source object:

1. Select the ProBindingSource control on the Design Canvas and either click its smart tag button (small right arrow) and click ProBindingSource Designer, or click ProBindingSource Designer at the bottom of the Properties view. The ProBindingSource Designer opens and displays the current schema definition.

2. Add or remove schema elements, or modify their properties, as explained in Manually defining the schema.

3. Click OK to save the schema definition.

**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.

**Related Concepts**
- Data-bound controls

**Related Tasks**
- Creating a binding source object
- Binding to an ABL data source
- Data binding example

**Related Reference**
- ProBindingSource Designer
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.

**Related Concepts**
- Data-bound controls

**Related Tasks**
- Creating a binding source object
- Editing a ProBindingSource control
- Editing a ProBindingSource control
- Data binding example

**Related Reference**
- ProBindingSource Designer
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 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).

Creating the data-displaying control

You must create a form and add a control capable of displaying data:

1. Create a new ABL Form container.
2. 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.)

Creating the binding source object

To create a binding source object and associate it with the grid control:

1. In the DB Structure view, expand the nodes of the SPORTS2000 database tree to show the columns in the Customer table.
2. Select the Name, City, and State columns. Use CTRL+Click to select multiple items.
3. 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 opens with the Customer table and the three columns that you selected appearing in the schema definition pane on the left.

4. Click OK to save the schema definition.

Note that columns labeled Name, City, and State appear in the grid control on the form.

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 its smart tag button (small right arrow) or click ProBindingSource Designer at the bottom of the Properties view.

Binding to the ABL data source

Define a query to retrieve the records for all customers who live in Texas:

1. Open the class file for the form in the OpenEdge ABL Editor by right-clicking on the form and selecting View Source.
2. 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:
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.

3. Save the class file.

Testing the finished control

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:
Related Concepts

Data-bound controls

Related Tasks

Creating a binding source object
Editing a ProBindingSource control
Editing a ProBindingSource control
Binding to an ABL data source

Related Tasks

Creating a binding source object
Creating a data-displaying control
Binding to an ABL data source
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. This section includes the following topics:

Creating event subscriptions
Coding event methods
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. OpenEdge Architect automatically opens the class file in the ABL 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>OpenEdge Architect>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.

**Related Concepts**

Event behavior
**Related Tasks**

- Coding event methods
- Using the Properties view

**Related Reference**

- Visual Designer Properties view
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:

```abl
METHOD PRIVATE VOID ultraButton1_Click ( sender AS System.Object, e AS System.EventArgs ) :

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>OpenEdge Architect>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.

**Related Concepts**

Event behavior

**Related Tasks**

Creating event subscriptions

**Related Reference**

Visual Designer Properties view
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. This section includes the following topics:

- Sharing Toolbox settings
- Managing control groups
- Managing controls
- Protecting Toolbox customizations
- Restoring default Toolbox settings
- Choosing the Toolbox docking position

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.

**Related Tasks**
- Managing control groups
- Managing controls
- Protecting Toolbox customizations
- Restoring default Toolbox settings
- Choosing the Toolbox docking position

**Related Reference**
- Visual Designer Editor
Managing control groups

As installed, the Toolbox has two or three control groups:

- Microsoft Controls
- OpenEdge Controls
- 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 OpenEdge Architect.

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.

Related Tasks

Sharing Toolbox settings
Managing controls
Protecting Toolbox customizations
Restoring default Toolbox settings
Choosing the Toolbox docking position

Related Reference

Visual Designer Editor
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.

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 OpenEdge Architect 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 OpenEdge Architect 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.

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.

Related Tasks

Sharing Toolbox settings
Managing control groups
Protecting Toolbox customizations
Restoring default Toolbox settings
Choosing the Toolbox docking position
Related Reference

Add Controls dialog
Visual Designer Editor
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.

**Related Tasks**
- Sharing Toolbox settings
- Managing control groups
- Managing controls
- Restoring default Toolbox settings
- Choosing the Toolbox docking position

**Related Reference**
- Visual Designer Editor
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.

**Related Tasks**
- Sharing Toolbox settings
- Managing control groups
- Managing controls
- Protecting Toolbox customizations
- Choosing the Toolbox docking position

**Related Reference**
- Visual Designer Editor
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 \textit{Window>Preferences>OpenEdge Architect>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.

\textbf{Related Tasks}
- Managing control groups
- Managing controls
- Protecting Toolbox customizations
- Restoring default Toolbox settings

\textbf{Related Reference}
- Visual Designer Editor
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
- Adding and removing assembly references
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.

**Related Concepts**
- Introducing the Class Browser

**Related Tasks**
- Getting control-specific information
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 Advanced GUI design, or where the code was manually developed outside the Visual Designer. OpenEdge Architect 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 Resources 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>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.

Related Concepts

.NET assemblies and the Visual Designer
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 OpenEdge Architect, 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 Architect menu or from a command prompt.
From OpenEdge Architect

From within OpenEdge Architect, 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 OpenEdge Architect if one or more projects in the current workspace use the updated assembly references.
- Recompile any projects that use the updated assembly references.

Related Concepts

**.NET assemblies and the Visual Designer**
Reference

This section includes the following topics:

Visual Designer Preferences page
OpenEdge GUI for .NET wizards
Workspace components
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>OpenEdge Architect>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 OpenEdge project properties page.)</td>
</tr>
<tr>
<td><strong>Show Smart-Tags</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>
<tr>
<td><strong>Show grid</strong></td>
<td>Enables or disables the visible display of the grid on forms on the Design Canvas.</td>
</tr>
<tr>
<td><strong>Snap to grid</strong></td>
<td>Enables or disables automatic alignment to the design grid when controls are placed or moved on the Design Canvas.</td>
</tr>
<tr>
<td><strong>Undo history size</strong></td>
<td>Specifies the number of actions that the Visual Designer can undo or redo.</td>
</tr>
</tbody>
</table>
Add untranslatable attribute to all Visual Designer-generated strings

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.

**Related Tasks**

- Working with UI controls
- Using built-in design tools
- Creating event subscriptions
- Sizing objects
- Positioning objects on a form
OpenEdge GUI for .NET wizards

This section includes the following topics:

- New ABL Form wizard
- New ABL Dialog wizard
- New ABL MDI Form wizard
- ProBindingSource Designer
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:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Package root</strong></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><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>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>
<tr>
<td><strong>Inherits</strong></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.<strong>) Note:</strong> 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.</td>
</tr>
</tbody>
</table>
### 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.

---

**Related Concepts**

- Visual container types

**Related Tasks**

- Developing a visual container
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:

<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>Dialog 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>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. <strong>Note:</strong> 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.</td>
</tr>
</tbody>
</table>
| **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. |

**Related Concepts**
- Visual container types

**Related Tasks**
- Developing a visual container
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>Package root</th>
<th>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).</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 Browse 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>MDI Form 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>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 Browse or enter the class path by typing. A period (.) must separate each node from its parent folder in the path name. <strong>Note:</strong> 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.</td>
</tr>
</tbody>
</table>
**Implements**

<table>
<thead>
<tr>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>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. <strong>Note:</strong> The class cannot implement a generic interface.</td>
<td></td>
</tr>
</tbody>
</table>

**Generate default constructor**

<table>
<thead>
<tr>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<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>
<td></td>
</tr>
</tbody>
</table>

**Generate destructor**

<table>
<thead>
<tr>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<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>
<td></td>
</tr>
</tbody>
</table>

**Generate super class constructors**

<table>
<thead>
<tr>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>If checked, specifies that the class is to include all constructor methods declared in the parent class.</td>
<td></td>
</tr>
</tbody>
</table>

**Add routine-level error handling**

<table>
<thead>
<tr>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>If checked, specifies that the ROUTINE-LEVEL ON ERROR UNDO, THROW statement will be included in the generated ABL code.</td>
<td></td>
</tr>
</tbody>
</table>

**Throw a Not Implemented exception**

<table>
<thead>
<tr>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<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>
<td></td>
</tr>
</tbody>
</table>

**Return a default value**

<table>
<thead>
<tr>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<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>
<td></td>
</tr>
</tbody>
</table>

**Description**

<table>
<thead>
<tr>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lets you optionally enter a description for the class. This text will appear in the file header.</td>
<td></td>
</tr>
</tbody>
</table>

**Purpose**

<table>
<thead>
<tr>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lets you optionally enter an explanation of the purpose of the class. This text will appear in the file header.</td>
<td></td>
</tr>
</tbody>
</table>

**Related Concepts**

- Visual container types

**Related Tasks**

- Developing a visual container
**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>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Package root</strong></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><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>User Control 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>
<tr>
<td><strong>Inherits</strong></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. <strong>Note:</strong> 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.</td>
</tr>
<tr>
<td><strong>Implements</strong></td>
<td>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. <strong>Note:</strong> The class cannot implement a generic interface.</td>
</tr>
</tbody>
</table>
### 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 superclass 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.

---

**Related Concepts**
- Custom user-defined controls

**Related Tasks**
- Creating a user control
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><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>Inherited Control 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>
<tr>
<td><strong>Inherits</strong></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. <strong>Note:</strong> 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.</td>
</tr>
</tbody>
</table>
| **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. |

**Related Concepts**

- Custom user-defined controls

**Related Tasks**

- Creating an inherited control
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 Tab
- 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>Tables (left) pane</th>
<th>Displays a hierarchical representation of the tables defined in the current schema.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fields (middle) pane</td>
<td>Displays the fields (columns) of the currently selected table.</td>
</tr>
<tr>
<td>Properties (right) pane</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 run time.</td>
</tr>
<tr>
<td>Add Table</td>
<td>Adds a table node. The new table is a child of the one above it in the tree.</td>
</tr>
<tr>
<td>Add Field</td>
<td>Adds a field node. The new field belongs to the table above it in the tree.</td>
</tr>
<tr>
<td>Delete</td>
<td>Deletes the selected node and its child nodes, if any.</td>
</tr>
<tr>
<td>Move Up</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>Move Down</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>Import from Database</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>Import from File</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>
</tbody>
</table>

Related Concepts

Data-bound controls
**Related Tasks**

- Linking controls to data
Workspace components

This section includes the following topics:

Visual Designer Editor
Visual Designer Properties view
Add Controls dialog
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, 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.

**Related Tasks**

- Using the Toolbox
- Choosing controls for the UI
- Defining properties and appearance
- Customizing the Visual Designer Toolbox
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).

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 Design Canvas, 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 Design Canvas, 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><img src="image" alt="Sort by Categories" /></td>
<td>Displays the grid with entries grouped in categories.</td>
</tr>
<tr>
<td><img src="image" alt="Sort Alphabetically" /></td>
<td>Displays the grid with entries in alphabetical order.</td>
</tr>
<tr>
<td><img src="image" alt="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>
Related Tasks

Using the Properties view
Coding event logic
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 OpenEdge Architect 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><strong>Control Group</strong></td>
<td>Lets you choose the Toolbox control group to which the selected controls are to be added.</td>
</tr>
<tr>
<td><strong>Filter</strong></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><strong>Clear</strong></td>
<td>Clears the Filter field and restores the control list to its unfiltered state.</td>
</tr>
<tr>
<td><strong>Browse</strong></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><strong>OK</strong></td>
<td>Closes the Add Controls dialog and adds all selected controls to the Toolbox.</td>
</tr>
<tr>
<td><strong>Cancel</strong></td>
<td>Closes the Add Controls dialog without adding any controls to the Toolbox.</td>
</tr>
</tbody>
</table>

**Related Tasks**

Managing controls
Introducing the Class Browser

The Class Browser is an OpenEdge Architect 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 Advanced GUI 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 OpenEdge Architect 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 OpenEdge Architect for the entire workspace. For details, see the description of Class Cache options in the OpenEdge Architect 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).

**Related Concepts**

- The Class Browser view
- External resources and the Class Browser

**Related Tasks**

- Accessing the Class Browser
- Viewing information in the Class Browser
- Adding external resources

**Related Reference**

- Browser pane
- Content pane
- Summary pane
- Class Browser toolbar
Concepts

This section describes the following Class Browser concepts:

- The Class Browser view
- External resources and the Class Browser

Related Tasks

Accessing the Class Browser
Viewing information in the Class Browser
Adding external resources

Related Reference

Browser pane
Content pane
Summary pane
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.

Related Tasks

Accessing the Class Browser
Viewing information in the Class Browser
Adding external resources

**Related Reference**

- Browser pane
- Content pane
- Summary pane
- Class Browser toolbar
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 OpenEdge Architect sessions.

If you close and then reopen the Class Browser during the same OpenEdge Architect 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.

Related Tasks
- Viewing information in the Class Browser
- Adding external resources

Related Reference
- Add External Resources dialog
Tasks

This section describes the following Class Browser tasks:

- Accessing the Class Browser
- Viewing information in the Class Browser
- Adding external resources
- Removing external resources
- Showing public, private, or inherited members
- Searching in the Class Browser
- Copying to the Clipboard

Related Concepts

- The Class Browser view
- External resources and the Class Browser

Related Reference

- Browser pane
- Content pane
- Summary pane
- Class Browser toolbar
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 AppServer, or OpenEdge Visual Designer perspective, click **Fast View**. Choose **Class Browser** from the list.
- From any OpenEdge Architect perspective, press **CTRL+F12**. Depending on the perspective, the Class Browser opens either as a Fast View (from the OpenEdge Editor, OpenEdge AppBuilder, OpenEdge AppServer, or OpenEdge Visual Designer) or as you have configured it as a view.
- From the **Workbench** menu, 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**.

**Related Concept**

The Class Browser view

**Related Task**

Viewing information in the Class Browser

**Related Reference**

Browser pane
Content pane
Summary pane
Class Browser toolbar
Viewing information in the Class Browser

You can use either of two modes to view information in the Class Browser:

<table>
<thead>
<tr>
<th>Browse by Resource</th>
<th>Displays all resources in an overall view. You are not required to have an OpenEdge project available to browse by resource.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browse by Project</td>
<td>Displays all resources by project. ABL built-in types andmscorlib.dll are also shown for each project node when you browse by project. <strong>Note:</strong> The Browse by Project mode does not display external resources.</td>
</tr>
</tbody>
</table>

To browse by resource, click **Browse by Resource** in the Class Browser toolbar; to browse by project, click **Browse by Project**.

**Related Concept**
- The Class Browser view

**Related Tasks**
- Browsing by resource
- Browsing by project

**Related Reference**
- Browser pane
- Content pane
- Summary pane
- Class Browser view icons
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.

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.

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.
- 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.

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.

**Related Concept**

The Class Browser view

**Related Task**

Browsing by project

**Related Reference**

Browser pane
Content pane
Summary pane
Class Browser view icons
Browsing by project

Browse by project to see a list of open OpenEdge Architect 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.

To begin, click **Browse by Project** in the Class Browser toolbar. A list of available OpenEdge Architect projects appears in the Browser pane, and the Content and Summary panes are empty.

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.

**Related Concept**

The Class Browser view

**Related Task**

Browsing by resource

**Related Reference**

Browser pane  
Content pane  
Summary pane  
Class Browser view icons
Showing public, protected, or inherited 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.

To choose which members you want the Class Browser to display:

1. 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.

**Related Concept**

The Class Browser view

**Related Task**

Browsing by resource
Browsing by project

**Related Reference**

Browser pane
Content pane
Summary pane
Class Browser view icons
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 OpenEdge Architect sessions.

When you add an external assembly reference, the Class Browser automatically updates the classBrowser.xml file for the workspace.

To add one or more external file resources:

1. 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. Select the appropriate tab for the resource you want to add:

   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.

   To add an assembly from the GAC:
   a. Choose the Global Assemblies tab.
   b. Select one or more (use CTRL+CLICK) assemblies to add.
   c. Click OK. The Add External Resources dialog closes. The resource or resources are added and appear in the Browser pane.

   To add a local assembly:
   a. Choose the Local Assemblies tab.
   b. Provide a path or locate the assembly file or executable file you want to add.
   c. Click OK.

Related Concept
External resources and the Class Browser

Related Task
Removing external resources
**Related Reference**

Add External Resources dialog
Removing external resources

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.

*Related Concept*

External resources and the Class Browser

*Related Task*

Adding external resources

*Related Reference*

Browser pane
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.

**Related Reference**

- Browser pane
Copy your 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.

**Related Concept**

The Class Browser view

**Related Task**

Viewing information in the Class Browser

**Related Reference**

Browser pane
Content pane
Summary pane
Reference

This section includes the following topics:

- Browser pane
- Content pane
- Summary pane
- Class Browser toolbar
- Class Browser view icons
- Add External Resources dialog

Related Concept

The Class Browser view

Related Tasks

Accessing the Class Browser
Adding external resources
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.

Related Concept
The Class Browser view

Related Tasks
- Browsing by resource
- Browsing by project
- Searching in the Class Browser

Related Reference
- Content pane
- Summary pane
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.

<table>
<thead>
<tr>
<th>When you select this item in the Browser pane . . .</th>
<th>The Content pane displays . . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>A project</td>
<td>The resources associated with the selected project</td>
</tr>
<tr>
<td>A resource</td>
<td>The packages or namespaces for the types contained in the selected resource</td>
</tr>
<tr>
<td>A package or a namespace</td>
<td>The types for the selected package or namespace within the resource</td>
</tr>
<tr>
<td>A type</td>
<td>The members of the selected type: data members, constructors, methods, properties, and events</td>
</tr>
</tbody>
</table>

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.

**Related Concept**

The Class Browser view

**Related Task**

Viewing information in the Class Browser

**Related Reference**

Browser pane  
Summary pane
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>
<tbody>
<tr>
<td>An assembly, procedure library, or path entry</td>
<td>▪ The name of the resource.</td>
</tr>
<tr>
<td></td>
<td>▪ The location of (path to) the resource.</td>
</tr>
<tr>
<td></td>
<td>▪ For an external resource, internal resource, or if selected by project: a list of project names that reference the resource.</td>
</tr>
<tr>
<td>▪ A summary (description) of the resource, if available.</td>
<td></td>
</tr>
<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>▪ The list of resources for which the package or namespace is a member.</td>
<td></td>
</tr>
<tr>
<td>▪ A summary (description) of the package or namespace.</td>
<td></td>
</tr>
<tr>
<td>A type</td>
<td>▪ The name of the type.</td>
</tr>
<tr>
<td>▪ Whether it is FINAL or ABSTRACT.</td>
<td></td>
</tr>
<tr>
<td>▪ The list of packages or namespaces for which the type is a member.</td>
<td></td>
</tr>
<tr>
<td>▪ The name of the base type it inherits from.</td>
<td></td>
</tr>
<tr>
<td>▪ The list of interfaces being implemented.</td>
<td></td>
</tr>
<tr>
<td>▪ A summary (description) of the type, if available.</td>
<td></td>
</tr>
<tr>
<td>▪ Examples of ABL syntax using the type.</td>
<td></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>
</tbody>
</table>

| A method | The name of the method and details about it, including PUBLIC, PROTECTED, FINAL, OVERRIDE, STATIC, or ABSTRACT. |
| | ABL syntax showing the method definition, which includes the access modifier, the method name, the method's parameters, and the return value. |
| | The name of the type that contains the method. |
| | A summary (description) of the method, if available. |
| Also shown if applicable: | |
| | A description of the return values of the method (if available). |
| | Parameters (if available). |

| 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). |

**Related Concept**

The Class Browser view

**Related Task**

Viewing information in the Class Browser

**Related Reference**

Browser pane  
Content pane
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>Button</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Navigate back through recently selected items.</td>
</tr>
<tr>
<td></td>
<td>Navigate forward through recently selected items.</td>
</tr>
<tr>
<td></td>
<td>Browse by resource.</td>
</tr>
<tr>
<td></td>
<td>Browse by project.</td>
</tr>
<tr>
<td></td>
<td>Add one or more external resources.</td>
</tr>
<tr>
<td></td>
<td>Collapse all expanded tree nodes in the Browser pane.</td>
</tr>
<tr>
<td></td>
<td>Refresh the Class Browser view.</td>
</tr>
<tr>
<td></td>
<td>Choose whether to show public members, show protected members, and/or show inherited members.</td>
</tr>
<tr>
<td></td>
<td>Minimize the Class Browser view.</td>
</tr>
<tr>
<td></td>
<td>Maximize the Class Browser view.</td>
</tr>
</tbody>
</table>

Related Concept
- The Class Browser view

Related Task
- Viewing information in the Class Browser
Related Reference

Browser pane
Content pane
Summary pane
## 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>![Icon]</td>
<td>Class Browser view.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Browse by resource.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Browse by project.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Add external resources.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Assembly.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Progress Procedure Library.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Path entry.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>ABL built-in types.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Search.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Clear search.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Class.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Interface.</td>
</tr>
<tr>
<td>![Icon]</td>
<td>Constructor tree.</td>
</tr>
</tbody>
</table>
| ![Icon] | Constructor.  
In this case, the green dot indicates that the constructor is public. See the following table for additional clarification. |
| ![Icon] | Method tree. |
| ![Icon] | Method. |
| ![Icon] | Event tree. |
| ![Icon] | Event. |
| ![Icon] | Data member tree. |
| ![Icon] | Data member. |
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:

<table>
<thead>
<tr>
<th>If you see this additional indicator...</th>
<th>The item is...</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>A green dot</td>
<td>Public</td>
<td><img src="img" alt="Capacity" /></td>
</tr>
<tr>
<td>A yellow diamond</td>
<td>Protected</td>
<td><img src="img" alt="TestMethod" /></td>
</tr>
<tr>
<td>The letter S</td>
<td>Static</td>
<td><img src="img" alt="Adapter(System.Collections.IList)" /></td>
</tr>
</tbody>
</table>
| The letter C                            | A constructor  | ![TestProperties()](img)  
**Note:** In this example, the green dot indicates that the item is also public. |
| The letter F                            | Final          | ![DictionaryBase](img) |
| A red box containing a white letter X  | Not available  | ![FormMenu(not available)](img) |

**Related Concept**

The Class Browser view

**Related Task**

Viewing information in the Class Browser

**Related Reference**

Browser pane  
Content pane  
Summary pane
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** 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>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-code path/PLs</td>
<td>A path or a Progress procedure library</td>
</tr>
<tr>
<td>Global Assemblies</td>
<td>An assembly from the Global Assembly Cache</td>
</tr>
<tr>
<td>Local Assemblies</td>
<td>An assembly or an executable file</td>
</tr>
</tbody>
</table>

**Related Concept**

External resources and the Class Browser

**Related Tasks**

Adding external resources
Removing external resources

**Related Reference**

Browser pane
Introducing launch configurations and the OpenEdge Architect Debugger

When you run or debug an ABL program, and when you start an AppServer instance, in OpenEdge Architect, you use a launch configuration that 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, OpenEdge Architect starts its Debugger.

The topics in this help module explain:

- How to manage and use launch configurations
- How to use the OpenEdge Architect Debugger

**Related Concepts**
- Launch configurations for running and debugging programs
- OpenEdge Architect Debugger

**Related Tasks**
- Using launch configurations
- Using the OpenEdge Architect Debugger

**Related Reference**
- Launch configuration settings
- Debugger dialogs
- Debugger views
Concepts

The topics in this section include:

Launch configurations for running and debugging programs
OpenEdge Architect Debugger
Launch configurations for running and debugging programs

You use a launch configuration to run or debug an ABL program or start an AppServer instance in the OpenEdge Architect environment. 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.

OpenEdge Architect provides a Configurations wizard that you can use 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. It includes the following topics:

- Accessing launch configuration functions
- Essential launch configuration settings
- How OpenEdge creates or reuses a configuration

**Launch configurations versus OpenEdge run configurations**

OpenEdge Architect versions up to and including 10.1C used run configurations that enabled ABL programs to execute outside the Eclipse framework. In later versions, launch configurations replace and extend the functionality formerly provided by run configurations.

OpenEdge Architect no longer supports run configurations. However, a migration procedure converts any existing run configurations found in the workspace to launch configurations. This migration occurs automatically and requires no user action.

**Note:** After completing the migration procedure, OpenEdge Architect deletes the original run configurations. Therefore, once converted, a run configuration is not accessible via earlier versions of OpenEdge Architect. Launch configurations are not backward-compatible with OpenEdge Architect versions earlier than 10.2A.
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:

![Run and Debug buttons](image)

- 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 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 instance (OpenEdge Architect does not support debugging of an AppServer instance). The Servers view buttons do not include the down arrow, so click the arrow on the main toolbar to open the menu.

**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 OpenEdge Application or Debug as OpenEdge Application** - 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

**Related Concepts**

OpenEdge Architect Debugger

**Related Tasks**

Using launch configurations
Related Reference

Launch configuration settings
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 Resources 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, How OpenEdge creates or reuses a configuration.

**Note:** These options do not apply to AppServer launch configurations.

**Related Concepts**
- Accessing launch configuration functions
- How OpenEdge creates or reuses a configuration

**Related Tasks**
- Using launch configurations

**Related Reference**
- Launch configuration settings
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>OpenEdge Application or Debug As>OpenEdge Application.)

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 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:

1. 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.

2. 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 an AppServer instance

To determine whether to use an existing configuration to start an AppServer instance, OpenEdge simply looks for a configuration specifying that server. If such a configuration exists, OpenEdge uses it to start the AppServer. (OpenEdge automatically creates a default launch configuration when each AppServer 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, OpenEdge creates and runs a new configuration with a name matching the server name.

Related Tasks

Using launch configurations
Using the OpenEdge Architect Debugger

Related Reference

Launch configuration settings
AppServer debugging

OpenEdge Architect does not support debugging of AppServer instances. To debug an AppServer, you must use the standalone OpenEdge Debugger.

You can start the AppServer from OpenEdge Architect and then switch to the standalone Debugger to debug it, but in such cases you must use the OpenEdge Architect Debug command (not the Start or Run command) to start the server. The Debug command sets the value of the debuggerEnabled property to TRUE in the ubroker.properties file. Starting the server with the Start or Run command sets this property to FALSE, thereby disabling debugging of the AppServer.

Other than the setting of the debuggerEnabled property, there is no functional difference between the Debug command and the Start or Run command for an AppServer instance.

Related Concepts
- Accessing launch configuration functions

Related Tasks
- Starting an AppServer instance

Related Reference
- OpenEdge AppServer settings
OpenEdge Architect Debugger

The OpenEdge Architect Debugger provides a full set of tools for testing and debugging ABL programs. The Debugger is an integral OpenEdge Architect component that works directly with the ABL Editor, making it easy to identify and fix problems as you work.

You can run multiple debugging sessions. Each session uses a dedicated instance of the OpenEdge runtime engine, providing these benefits:

- You can debug multiple programs simultaneously, whether the programs are in the same or different projects.
- You can debug multiple instances of the same program.
- You can edit the source code with full use of all Editor features while a debugging session is active.

This section includes the following topics:

- The OpenEdge Debugger perspective
- Prerequisites for using the Debugger
- OpenEdge Debugger preferences
- Debugging operations

The standalone OpenEdge Debugger

In addition to the OpenEdge Architect Debugger, OpenEdge provides a standalone Debugger. You can determine which Debugger will be of greater use to you.

Use the OpenEdge Architect Debugger if:

- You are working in OpenEdge Architect.
- You are designing classes.
- You do not need to do remote debugging.

Use the standalone OpenEdge Debugger if:

- You want to attach to a running process.
- You want to do remote debugging.
- You want to do WebSpeed debugging.

To launch the standalone OpenEdge Debugger, select Debugger from the 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 Web site.

For the purposes of this online help, however, all references to "the OpenEdge Debugger" pertain to the OpenEdge Architect Debugger.
The OpenEdge Debugger perspective

In its default state, the OpenEdge Debugger perspective partitions the Workbench into four functional areas:

- **Program execution area** - Contains the **Debug** view, which displays a session tree with a stack trace 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.

**Related Concept**

Debugging operations

**Related Tasks**

Using the OpenEdge Architect Debugger

**Related Reference**

Debugger dialogs
Debugger views
Prerequisites for using the Debugger

Debugging is automatically enabled for you in OpenEdge Architect. 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 build properties 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 Adding existing database connections to a project.

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.

Related Tasks

- Enabling debugging
- Configuring r-code generation
OpenEdge Debugger preferences

There are two preferences you can optionally use to better manage your Debugger environment.

Enable property evaluation

The Enable property evaluation preference allows you 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.)

Connection timeout

The Connection timeout value indicates how long, in milliseconds, OpenEdge Architect 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).

Related Task

Setting Debugger preferences

Related Reference

Debug Preferences dialog
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. These operations are described in the following topics:

Options for controlling program execution
Options for monitoring results
Options for modifying code while debugging
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 (other than in an include file). 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 at any time.

These features are more fully described in the following topics:

Breakpoints
Code-stepping
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. 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.

If the expression attempts to execute an I/O blocking statement (for example, a user-defined function that includes an UPDATE statement), the Debugger displays an error message. In this case, code in the user-defined function preceding the UPDATE statement is executed.

**Related Concepts**

Code-stepping
**Related Task**
- Using breakpoints

**Related Reference**
- Breakpoints view
**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 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 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.

- **Terminate** - Interrupts execution and ends the Debugger session.

**Garbage collection in the 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.
Running and Debugging ABL Programs

Related Concept

Breakpoints

Related Tasks

Stepping through the code
Using the Debug view and stack trace

Related Reference

Debug view
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.

**Related Concept**

The OpenEdge Debugger perspective

**Related Task**

Monitoring program execution

**Related Reference**

Debugger views
Options for modifying code 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.

Related Concept
- Options for monitoring results

Related Task
- Changing program code

Related Reference
- Set Value dialog
- Variables view
Tasks

This section includes the following topics:

- Using launch configurations
- Using the OpenEdge Architect Debugger
Using launch configurations

You can run and debug your ABL programs directly from within the OpenEdge Architect environment. Each time you do so, the program runs in a launch configuration, which you can customize to suit the needs of your application and its target implementation.

Related Concepts
Launch configurations for running and debugging programs

Related Tasks
Running or debugging a program
Managing launch configurations

Related Reference
Launch configuration settings
Running or debugging a program

To run or debug an ABL program (.p, .w, or .cls file) that exists in an OpenEdge Architect 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

With 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 Resources view, or open it in the ABL Editor.

   **Note:** You cannot 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>OpenEdge Application** or **Debug As>OpenEdge Application**.
   - Right-click in the ABL Editor window and select **Run As>OpenEdge Application** 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>- This is the quickest way to run a program.</td>
<td>- The configuration that OpenEdge chooses or creates may not meet all of your needs.</td>
</tr>
<tr>
<td>- 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>- 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>- 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>
**Selecting an existing launch configuration**

With this technique, you choose a launch configuration that you have previously defined.

1. Select the file in the Resources 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.
   - 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.
   - 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 (under the OpenEdge Application node) 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 for information on how to proceed with the debugging session.

**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.

**Related Concepts**

Launch configurations for running and debugging programs

**Related Tasks**

Defining a launch configuration  
Creating project-standard and program-specific configurations  
Adding a configuration to the menu  
Copying and deleting launch configurations  
Setting launch configuration preferences

**Related Reference**

Launch configuration settings
Starting an AppServer instance

To start an AppServer instance, first select the instance in the Servers view, which is part of the default OpenEdge AppServer 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 launch configuration.
- Explicitly define and run a new AppServer launch configuration.

Note: You cannot use the OpenEdge Architect Debugger to debug an AppServer instance. See AppServer debugging for more information.

Letting OpenEdge start the AppServer 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 starts in the configuration chosen or created by OpenEdge.

Selecting an existing AppServer launch configuration

With this technique, you choose a launch configuration that was previously defined.

Click the down arrow on the Run button or the Debug button on the main toolbar to open the drop-down menu.

- 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.
- 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 (under the OpenEdge AppServer node) and then click the Run button or the Debug button in the lower right corner.

The AppServer starts in the selected configuration.

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 menu or the Debug menu if you wish.

4. Run the configuration either from the Configurations wizard or from the Run or Debug menu.

Related Concepts
- AppServer debugging
- Launch configurations for running and debugging programs

Related Tasks
- Managing launch configurations
- Adding a configuration to the menu

Related Reference
- OpenEdge AppServer settings
Managing launch configurations

You use a launch configuration every time you run or debug an ABL program in OpenEdge Architect. The following topics in this section provide instructions for working with launch configurations:

- Defining a launch configuration
- Creating project-standard and program-specific configurations
- Adding a configuration to the menu
- Copying and deleting launch configurations
- Setting launch configuration preferences

**Related Concepts**

Launch configurations for running and debugging programs

**Related Tasks**

Running or debugging a program

**Related Reference**

Launch configuration settings
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. To create a new launch configuration, click on the wizard toolbar and enter a name for the configuration in the **Name** field.

   To edit an existing configuration, select it in the tree view in the left pane.

3. The wizard includes tabs labeled **Main**, **Startup**, **PROPATH**, **Databases**, **ABL**, **Culture**, **Performance**, **Logging**, **Environment**, and **Common**. 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.

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 OpenEdge Application** command.

- **Debug configurations** - The OpenEdge Architect 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 runtime** 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 runtime** on the Main tab.
Related Concepts

Launch configurations for running and debugging programs

Related Tasks

Running or debugging a program
Creating project-standard and program-specific configurations
Adding a configuration to the menu
Copying and deleting launch configurations
Setting launch configuration preferences

Related Reference

Launch configuration settings
Creating project-standard and program-specific configurations

The quickest way to run or debug a program is to use the **Run/Debug as OpenEdge Application** 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, OpenEdge 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 OpenEdge Application** 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.

**Related Concepts**
- Launch configurations for running and debugging programs

**Related Tasks**
- Running or debugging a program
- Defining a launch configuration
Adding a configuration to the menu
Copying and deleting launch configurations
Setting launch configuration preferences

Related Reference
Launch configuration settings
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.

**Related Concepts**
- Launch configurations for running and debugging programs

**Related Tasks**
- Running or debugging a program

**Related Reference**
- Configurations wizard
Copy the 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 **Duplicate** 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 **Delete** on the wizard toolbar, or press Delete.

Alternatively, right-click a configuration in the left pane and select **Duplicate** or **Delete** from the context menu.

**Related Concepts**
Launch configurations for running and debugging programs

**Related Tasks**
Defining a launch configuration
Creating project-standard and program-specific configurations
Adding a configuration to the menu
Setting launch configuration preferences

**Related Reference**
Launch configuration settings
Setting launch configuration preferences

Both OpenEdge Architect and the Eclipse framework offer options that affect launch configurations.

**OpenEdge Architect launching preferences**

To set OpenEdge Architect launching preferences, select **Window>Preferences>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**.

**Related Concepts**

Launch configurations for running and debugging programs

**Related Tasks**

Defining a launch configuration
Creating project-standard and program-specific configurations

**Related Reference**

Main tab (Application)
Using the OpenEdge Architect Debugger

Demo
Introducing the OpenEdge Architect Debugger

This section includes the following topics:

- Setting Debugger preferences
- Enabling debugging
- Launching the Debugger
- Configuring r-code generation
- Controlling program flow
- Monitoring program execution
- Changing program code
- Closing a Debugger session
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 > OpenEdge Architect > 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.)

   - **Connection timeout (ms)** - Provide the time-out value that you want the OpenEdge Architect 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**.

**Related Concept**

OpenEdge Debugger preferences

**Related Reference**

Debug preferences dialog
Enabling debugging

Debugging is automatically enabled for you in OpenEdge Architect. 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 **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.

**Related Concept**

Prerequisites for using the Debugger

**Related Task**

Launching the Debugger
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.

**Note:** You cannot 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 (้อย) 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.

**Related Concepts**

Options for controlling program execution
Launch configurations for running and debugging programs

**Related Tasks**

Enabling debugging
Configuring r-code generation
Controlling program flow
Monitoring program execution
Closing a Debugger session
Defining a launch configuration
Copying and deleting launch configurations
Setting launch configuration preferences
Managing launch configurations
Running or debugging a program

**Related Reference**

Debugger dialogs
Debugger views
Launch configuration settings
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 Resources view, select your project and open its Properties window (right-click and select Properties, or select File>Properties from the menu bar).
2. Select 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.

**Related Concept**
- Prerequisites for using the Debugger

**Related Task**
- Defining a launch configuration

**Related Reference**
- Build properties page
- PROPATH and Source properties page
Controlling program flow

The topics in this section include:

Using breakpoints
Stepping through the code
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.

This section discusses the following topics:

- Setting breakpoints
- Selectively enabling breakpoints
- Managing breakpoints
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.
- **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, 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.

Setting an at-line breakpoint

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.

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:

1. Open the **New Breakpoint** dialog by right-clicking in the **Breakpoints** view and selecting **Add Breakpoint**.
2. Select the **At Line** tab.
3. 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.
4. Enter the line number. You can leave the **Line Number** field blank to add the breakpoint at the first line of the specified procedure.
5. Optionally, enter a **condition statement**. The Debugger will suspend execution at the breakpoint only if the condition is true at that time.

**Note:** A breakpoint must be at an executable statement. If you try to set a breakpoint in an ineligible location, the breakpoint is inserted at the next eligible line.

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.

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 OpenEdge Architect Debugger sessions.

Breakpoints and include files

OpenEdge Architect does not support breakpoints within include files. Instead, set a
breakpoint, if desired, at the reference to the include file in the calling procedure.

**Setting an on-error breakpoint**

To define a breakpoint that takes effect when the program encounters an error:

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.

**Setting a watchpoint**

To define a breakpoint that takes effect when the value of a given variable, buffer field, object attribute, or object reference changes:

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.
   - Select the **Add Watch** option if you want to show this expression in the **Expressions view**.
4. Optionally, enter a **condition statement**. The Debugger will suspend execution only if the condition is true when the value changes for the specified expression.

**Related Concept**

- Breakpoints

**Related Tasks**

- Selectively enabling breakpoints
- Managing breakpoints
- Using watch expressions

**Related Reference**

- Breakpoints view
- New Breakpoint dialog
- Expressions view
Selectively enabling breakpoints

The **Breakpoints** view lists all breakpoints that are defined for the current 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.

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.

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.)

**Related Concept**

Breakpoints

**Related Tasks**

Setting breakpoints  
Managing breakpoints

**Related Reference**

Breakpoints view
Managing breakpoints

The **Breakpoints** view lists all breakpoints defined for the current workspace. You can edit or delete any breakpoint in the list.

**Editing breakpoints**

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.

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** for more information.

**Deleting breakpoints**

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**.

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**.

**Sharing breakpoints**

Use the **Export** and **Import** options on the **Breakpoints** view context (right-click) menu to share breakpoint definitions with other users. See **Importing breakpoints** for more information.

**Related Concept**

**Breakpoints**

**Related Task**

**Setting breakpoints**

**Related Reference**

**Breakpoints view**
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.

- **Terminate** - Interrupts execution and ends the Debugger session.

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.

**Garbage collection**

For details about the behavior of code-stepping with regard to garbage collection, see Garbage collection in the Debugger.

**Related Concepts**

- Code-stepping
- Breakpoints

**Related Tasks**

- Using the Debug view and stack trace
- Using breakpoints

**Related Reference**

- Debug view
Monitoring program execution

The topics in this section include:

- Using the Debug view and stack trace
- Using the Variables view
- Using the Variable Details dialog
- Using watch expressions
- Using the Dynamic Objects view
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 a session tree. As shown in the example below, the 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 that 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.

Related Tasks
- Stepping through the code
- Changing program code
- Launching the Debugger

Related Reference
- Debug view
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>Variables.

Selecting Variables view display information

To select the information you want to see in the view:

1. Click the drop-down menu from the Variables view toolbar.
2. Choose Layout>Show Columns. (You cannot select specific columns until you select the Show Columns option.)
3. Choose Layout>Select Columns. The Select Columns dialog appears.
4. 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 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 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 expand the hierarchy and show the subordinate values, click the plus symbol. You can collapse the hierarchy selectively by clicking the symbol, now a minus sign (-); or collapse all entries and show only top-level data elements by clicking Collapse All on the toolbar.

Note that 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

You can evaluate a property from the Variables view. 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 display

You can filter the display in the Variables view to show only selected element types. 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.

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

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, 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, enter the replacement value, and click OK.

Note: 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

You can add a watch expression for one or more elements so that you can track their values in the Expressions view. Select the desired entries, 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. You can 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.

Open the Variable Details dialog by either of the following methods:

- 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.

**Related Concepts**

Options for monitoring results
The OpenEdge Debugger perspective

**Related Tasks**

Using the Debug view and stack trace
Using the Variable Details dialog
Changing program code
Using watch expressions

**Related Reference**

Variables view
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 **Variable Details** dialog, you must have an active Debugger session and have data displayed in the **Variables** view. Do either of the following:

- 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 programmatically 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.

Related Concept
Options for monitoring results

Related Task
Using the Variables view
Changing program code
Using watch expressions

Related Reference
Variable Details dialog
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.

Adding watch expressions

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>Expression</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>_PCONTROL</td>
<td>Control number</td>
</tr>
<tr>
<td>_SERIAL-NUM</td>
<td>Serial number</td>
</tr>
<tr>
<td>COM</td>
<td>Command</td>
</tr>
<tr>
<td>CURRENT-LANGUAGE</td>
<td>Current language</td>
</tr>
<tr>
<td>DATASERVERS</td>
<td>Database servers</td>
</tr>
<tr>
<td>DBNAME</td>
<td>Database name</td>
</tr>
<tr>
<td>ETIME</td>
<td>Execution time</td>
</tr>
<tr>
<td>FRAME-DB</td>
<td>Frame database</td>
</tr>
<tr>
<td>FRAME-FIELD</td>
<td>Frame field</td>
</tr>
<tr>
<td>FRAME-INDEX</td>
<td>Frame index</td>
</tr>
<tr>
<td>FRAME-NAME</td>
<td>Frame name</td>
</tr>
<tr>
<td>FRAME-VALUE</td>
<td>Frame value</td>
</tr>
<tr>
<td>GO-PENDING</td>
<td>Go pending</td>
</tr>
<tr>
<td>LASTKEY</td>
<td>Last key</td>
</tr>
<tr>
<td>MACHINE-CLASS</td>
<td>Machine class</td>
</tr>
<tr>
<td>MESSAGE-LINES</td>
<td>Message lines</td>
</tr>
<tr>
<td>NUM-ALIASES</td>
<td>Number of aliases</td>
</tr>
<tr>
<td>NUM-DBS</td>
<td>Number of databases</td>
</tr>
<tr>
<td>OPSYS</td>
<td>Operating system</td>
</tr>
<tr>
<td>OS-DRIVES</td>
<td>Operating system drives</td>
</tr>
<tr>
<td>OS-ERROR</td>
<td>Operating system error</td>
</tr>
<tr>
<td>PROGRESS</td>
<td>Progress</td>
</tr>
<tr>
<td>PROMSGS</td>
<td>Prompts</td>
</tr>
<tr>
<td>PROPATH</td>
<td>Prompt path</td>
</tr>
<tr>
<td>PROVERSION</td>
<td>Prompt version</td>
</tr>
<tr>
<td>RETRY</td>
<td>Retry</td>
</tr>
<tr>
<td>RETURN-VALUE</td>
<td>Return value</td>
</tr>
<tr>
<td>SCREEN-LINES</td>
<td>Screen lines</td>
</tr>
<tr>
<td>SELF</td>
<td>Self</td>
</tr>
<tr>
<td>TIME</td>
<td>Time</td>
</tr>
<tr>
<td>TODAY</td>
<td>Today</td>
</tr>
<tr>
<td>TRANSACTION</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 <strong>Expressions</strong> 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>
<tr>
<td>Change Value</td>
<td>Lets you enter a new value for an element.</td>
</tr>
<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 <strong>Expressions</strong> 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>

### Related Concept

*Options for monitoring results*
Related Tasks

   Setting breakpoints
   Using the Variables view

Related Reference

   Expressions view
Using the Dynamic Objects view

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.

### 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 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 the preceding section.

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.

**Related Concept**
Options for monitoring results

**Related Reference**
Debug view
Dynamic Objects view
Filter Objects dialog
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 OpenEdge Debugger 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 OpenEdge Debugger 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.

Related Tasks

- Using the Variables view
- Using the Variable Details dialog
Closing a Debugger session

At any time that 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.

Related Tasks

Using the Debug view and stack trace
Stepping through the code
Reference

This section includes the following topics:

Debugger views
Debugger dialogs
Launch configuration settings

The topics in this section describe the available settings and controls related to defining and using launch configurations. The topics include:

- Configurations wizard
- Launching preferences
Configurations wizard

The topics in this section describe the options and controls available for defining launch configurations in the Configurations wizard. These topics are grouped separately in the following sections:

OpenEdge Application settings
OpenEdge AppServer settings
OpenEdge Application settings

The topics in this section describe the options and controls available for defining OpenEdge Application launch configurations in the Configurations wizard. There is a separate topic for each of the following tabs in this section of the wizard:

- Main
- Startup
- PROPATH
- Databases
- ABL
- Culture
- Performance
- Logging
- Environment
- Common
Main tab (Application)

The **Main** tab in the 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.

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.

Also see help for the other tabs of the Configurations wizard: **Startup**, **PROPATH**, **Databases**, **ABL**, **Culture**, **Performance**, **Logging**, **Environment**, and **Common**, as well as the settings for **OpenEdge AppServer launch configurations**.

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.

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><strong>Name</strong> <em>(accessible from all tabs)</em></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><strong>Project</strong></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 <strong>How OpenEdge creates or reuses a configuration</strong> for information about how this selection affects launch configuration behavior.</td>
<td></td>
</tr>
<tr>
<td><strong>Run this program/Run selected program in workspace</strong></td>
<td>Controls whether the configuration runs the program that currently has focus (in the Resources view or in the ABL Editor), or the specific program identified in the <strong>Startup program</strong> field.</td>
<td></td>
</tr>
<tr>
<td><strong>Startup program</strong></td>
<td>(Required if <strong>Run this program</strong> is selected) Specifies the procedure that executes when the configuration runs.</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>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. <strong>Note:</strong> A configuration that is used for debugging must start a new AVM. Debugging under the project AVM is not supported.</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 OpenEdge Architect.</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 running 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 <strong>ABL Virtual Machine</strong> section on this tab. <strong>Note:</strong> It is possible for the configuration to get into an infinite loop of restart cycles. In such a case, edit the configuration definition, deselect this option, and save the configuration.</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>
</tbody>
</table>
### Related Concepts
Launch configurations for running and debugging programs

### Related Tasks
Using launch configurations

### Related Reference
- Startup tab (Application)
- PROPATH tab (Application)
- Databases tab (Application)
- ABL tab (Application)
- Culture tab (Application)
- Performance tab (Application)
- Logging tab (Application)
- Environment tab (Application)
- Common tab (Application)

<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
<th>Related parameter (if applicable)</th>
</tr>
</thead>
<tbody>
<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>
<tr>
<td>Run or Debug</td>
<td>Runs the specified startup program or opens it in the OpenEdge Architect Debugger.</td>
<td></td>
</tr>
<tr>
<td>Close</td>
<td>Closes the Configurations wizard.</td>
<td></td>
</tr>
</tbody>
</table>
Startup tab (Application)

The **Startup** tab in the OpenEdge Application section of the Configurations wizard contains many of the most commonly used startup parameters for an ABL application launch configuration.

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.

Also see help for the other tabs in this section of the Configurations wizard: **Main**, **PROPATH**, **Databases**, **ABL**, **Culture**, **Performance**, **Logging**, **Environment**, and **Common**, as well as the settings for OpenEdge AppServer launch configurations.

Various options have the effect of invoking a startup parameter for the AVM session. 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.

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></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 <code>-param</code> startup parameter when the configuration starts.</td>
<td>Parameter (-p)</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>Initialization File (-ininame)</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 <strong>Main</strong> tab) for the assemblies directory and for loading of the Common Language Runtime.</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><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 <strong>Use project settings</strong> 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>
<tr>
<td><strong>Enable Progress Dynamics</strong></td>
<td>If checked, enables use of the Dynamics framework under the configuration. Also:</td>
<td>Dynamics Parameter (-icfparam)</td>
</tr>
<tr>
<td></td>
<td>▪ Specify icfstart.p as the startup program on the Main tab.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>▪ Include the -icfparam parameter in the Additional startup parameters field below.</td>
<td></td>
</tr>
<tr>
<td><strong>Configuration file</strong></td>
<td>(Required if <strong>Enable Progress Dynamics</strong> is selected) Specifies the full or relative (to the working directory) path to the Dynamics icfconfig.xml file.</td>
<td></td>
</tr>
<tr>
<td><strong>Session type</strong></td>
<td>Specifies the Dynamics session type to be used under the configuration.</td>
<td></td>
</tr>
<tr>
<td><strong>Additional startup parameters</strong></td>
<td>Optionally specifies any desired parameters (with values) not included elsewhere in this configuration definition. Do not include the –p parameter in this field.</td>
<td></td>
</tr>
<tr>
<td><strong>Apply</strong></td>
<td>Saves the current launch configuration definition.</td>
<td></td>
</tr>
<tr>
<td><strong>Revert</strong></td>
<td>Discards unsaved changes to the current launch configuration definition.</td>
<td></td>
</tr>
</tbody>
</table>
## Running and Debugging ABL Programs

### Related Concepts
Launch configurations for running and debugging programs

### Related Tasks
Using launch configurations

### Related Reference
Main tab (Application)
PROPATH tab (Application)
Databases tab (Application)
ABL tab (Application)
Culture tab (Application)
Performance tab (Application)
Logging tab (Application)
Environment tab (Application)
Common tab (Application)
Launching preferences

### Control

<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
<th>Related parameter (if applicable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run or Debug</td>
<td>Runs the specified startup program or opens it in the OpenEdge Architect Debugger.</td>
<td></td>
</tr>
<tr>
<td>Close</td>
<td>Closes the Configurations wizard.</td>
<td></td>
</tr>
</tbody>
</table>

**Run or Debug**

Runs the specified startup program or opens it in the OpenEdge Architect Debugger.

**Close**

Closes the Configurations wizard.
PROPATH tab (Application)

The PROPATH tab in the OpenEdge Application section of the Configurations wizard lets you control where OpenEdge searches for files and procedures when the launch configuration runs.

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.

Also see help for the other tabs in this section of the Configurations wizard: Main, Startup, Databases, ABL, Culture, Performance, Logging, Environment, and Common, as well as the settings for OpenEdge AppServer launch configurations.

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>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>Tree view</td>
<td>Shows the current PROPATH definition.</td>
</tr>
<tr>
<td>Add Project</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>Copy ini Settings</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>Other command buttons</td>
<td>Function as described in the PROPATH help for the Project Properties dialog.</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>Runs the specified startup program or opens it in the OpenEdge Architect Debugger.</td>
</tr>
<tr>
<td>Close</td>
<td>Closes the Configurations wizard.</td>
</tr>
</tbody>
</table>
Running and Debugging ABL Programs

**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.

**Related Concepts**
- Launch configurations for running and debugging programs
- AppServer debugging

**Related Tasks**
- Using launch configurations

**Related Reference**
- Main tab (Application)
- Databases tab (Application)
- Culture tab (Application)
- Performance tab (Application)
- Logging tab (Application)
- Environment tab (Application)
- Common tab (Application)
- Launching preferences
Databases tab (Application)

The **Databases** tab in the OpenEdge Application section of the Configurations wizard lets you specify the OpenEdge database connections used by an ABL application launch configuration.

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.

Also see help for the other tabs in this section of the Configurations wizard: **Main**, **Startup**, **PROPATH**, **ABL**, **Culture**, **Performance**, **Logging**, **Environment**, and **Common**, as well as the settings for OpenEdge AppServer launch configurations.

The **Databases** tab functions like the Database Connections page 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 <strong>Main</strong> 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 <strong>Main</strong> tab for the configuration.</td>
</tr>
<tr>
<td><strong>Database connection list</strong></td>
<td>Lists available OpenEdge database connections, filtered according to the <strong>Show selected/Show all/Show from project</strong> 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 Database Connections help 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>Discards unsaved changes to the current launch configuration definition.</td>
</tr>
<tr>
<td>Control</td>
<td>Purpose</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Run or Debug</td>
<td>Runs the specified startup program or opens it in the OpenEdge Architect Debugger.</td>
</tr>
<tr>
<td>Close</td>
<td>Closes the Configurations wizard.</td>
</tr>
</tbody>
</table>

**Related Concepts**
- Launch configurations for running and debugging programs
- Database connection profiles

**Related Tasks**
- Using launch configurations

**Related Reference**
- Main tab (Application)
- Startup tab (Application)
- PROPATH tab (Application)
- ABL tab (Application)
- Culture tab (Application)
- Performance tab (Application)
- Logging tab (Application)
- Environment tab (Application)
- Common tab (Application)
- Launching preferences
- Database Connections preference page
ABL tab (Application)

The ABL tab in the OpenEdge Application section of the Configurations wizard lets you set various options affecting ABL behavior of a launch configuration.

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.

Also see help for the other tabs in this section of the Configurations wizard: Main, Startup, PROPATH, Databases, Culture, Performance, Logging, Environment, and Common, as well as the settings for OpenEdge AppServer launch configurations.

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.

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 (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>
<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 Help 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>Debug Alert (debugalert)</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><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 NO-LOCK option.)</td>
<td>Reread Nolock (–rereadnolock)</td>
</tr>
<tr>
<td><strong>Use error stack</strong></td>
<td>If checked, allows error objects to save the ABL call stack in the CallStack property.</td>
<td>Error Stack (–errorstack)</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>Use Widget ID (–usewidgetid)</td>
</tr>
<tr>
<td><strong>Expand browse columns</strong></td>
<td>If checked, causes the AVM to turn on the EXPANDABLE and FIT-LAST-COLUMN options for browses created in the session.</td>
<td>Expand Browse (–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>Filtering Asynchronous COM Events (–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>Number of Databases (–h)</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>Nested Blocks (–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>Input Characters (–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>Token (–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>
### Control

<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
<th>Related parameter (if applicable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use dictionary expressions</td>
<td>If checked, causes ABL to use dictionary and help validation for all fields in all frames when you compile an application.</td>
<td>Dictionary Expressions (-dictexps)</td>
</tr>
<tr>
<td>Keyword forget file</td>
<td>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.</td>
<td>Keyword Forget List (-k)</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>
<tr>
<td>Run or Debug</td>
<td>Runs the specified startup program or opens it in the OpenEdge Architect Debugger.</td>
<td></td>
</tr>
<tr>
<td>Close</td>
<td>Closes the Configurations wizard.</td>
<td></td>
</tr>
</tbody>
</table>

**Related Concepts**

Launch configurations for running and debugging programs

**Related Tasks**

Using launch configurations

**Related Reference**

Main tab (Application)
Startup tab (Application)
PROPATH tab (Application)
Databases tab (Application)
Culture tab (Application)
Performance tab (Application)
Logging tab (Application)
Environment tab (Application)
Common tab (Application)
Culture tab (Application)

The Culture tab in the OpenEdge Application section of the Configurations wizard lets you set various internationalization options affecting ABL behavior of a launch configuration.

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.

Also see help for the other tabs of the Configurations wizard: Main, Startup, PROPATH, Databases, ABL, Performance, Logging, Environment, and Common, as well as the settings for OpenEdge AppServer launch configurations.

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.

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>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 the values in the startup parameter file(s) used by the configuration.</td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>Optionally specifies the initial return value for the CURRENT-LANGUAGE function.</td>
<td>Language (-l&lt;ng&gt;)</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>Control</td>
<td>Purpose</td>
<td>Related parameter (if applicable)</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------------------------------------------------------</td>
<td>------------------------------------</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>
<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 (-cpterm)</td>
</tr>
<tr>
<td>Stream code page</td>
<td>Specifies the Stream ccode 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 ccode 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>Use OS locale</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>
</tbody>
</table>
### Control

<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
<th>Related parameter (if applicable)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Date format</strong></td>
<td>Specifies the date-display format used by the configuration.</td>
<td>Date Format (-d)</td>
</tr>
<tr>
<td><strong>Year offset</strong></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><strong>Fractional separator</strong></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><strong>Thousands separator</strong></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><strong>Use 4-digit year</strong></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><strong>Apply</strong></td>
<td>Saves the current launch configuration definition.</td>
<td></td>
</tr>
<tr>
<td><strong>Revert</strong></td>
<td>Discards unsaved changes to the current launch configuration definition.</td>
<td></td>
</tr>
<tr>
<td><strong>Run or Debug</strong></td>
<td>Runs the specified startup program or opens it in the OpenEdge Architect Debugger.</td>
<td></td>
</tr>
<tr>
<td><strong>Close</strong></td>
<td>Closes the Configurations wizard.</td>
<td></td>
</tr>
</tbody>
</table>

### Related Concepts

Launch configurations for running and debugging programs

### Related Tasks

Using launch configurations

### Related Reference

Main tab (Application)
Startup tab (Application)
PROPATH tab (Application)
Databases tab (Application)
ABL tab (Application)
Performance tab (Application)
Logging tab (Application)
Environment tab (Application)
Common tab (Application)
Performance tab (Application)

The **Performance** tab in the OpenEdge Application section of the Configurations wizard lets you set various options affecting performance of applications running under a launch configuration.

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.

Also see help for the other tabs of the Configurations wizard: Main, Startup, PROPATH, Databases, ABL, Culture, Logging, Environment, and Common, as well as the settings for OpenEdge AppServer launch configurations.

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.

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><strong>Name</strong> <em>(accessible from all tabs)</em></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><strong>Use defaults</strong></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><strong>Temporary files directory</strong></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><strong>Enforce startup parameter limits</strong></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>Control</td>
<td>Purpose</td>
<td>Related parameter (if applicable)</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------------------------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td><strong>Schema field cache size</strong></td>
<td>Optionally specifies the number of entries in the schema field cache.</td>
<td>Schema Field Cache Size ((-fc))</td>
</tr>
<tr>
<td><strong>Record buffer size</strong></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><strong>Stack size</strong></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><strong>Local buffer size</strong></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><strong>Directory size</strong></td>
<td>Optionally specifies the number of compiled procedure directory entries.</td>
<td>Directory Size ((-D))</td>
</tr>
<tr>
<td><strong>Stash blocks</strong></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><strong>Temp-table buffer pool size</strong></td>
<td>Optionally specifies the number of buffers in the temporary table database pool.</td>
<td>Number of Buffers for Temporary Tables ((-Bt))</td>
</tr>
<tr>
<td><strong>Speed sort block size</strong></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><strong>Speed sort merge streams</strong></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><strong>Async queue size</strong></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))</td>
</tr>
<tr>
<td><strong>Maximum memory</strong></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>
Running 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 (-pls)</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>
<tr>
<td>Run or Debug</td>
<td>Runs the specified startup program or opens it in the OpenEdge Architect Debugger.</td>
<td></td>
</tr>
<tr>
<td>Close</td>
<td>Closes the Configurations wizard.</td>
<td></td>
</tr>
</tbody>
</table>

**Related Concepts**
Launch configurations for running and debugging programs

**Related Tasks**
Using launch configurations

**Related Reference**
- Main tab (Application)
- Startup tab (Application)
- PROPATH tab (Application)
- Databases tab (Application)
- ABL tab (Application)
- Culture tab (Application)
- Logging tab (Application)
- Environment tab (Application)
- Common tab (Application)
Logging tab (Application)

The **Logging** tab in the OpenEdge Application section of the Configurations wizard lets you set various options affecting logging of ABL applications running under a launch configuration.

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.

Also see help for the other tabs of the Configurations wizard: **Main**, **Startup**, **PROPATH**, **Databases**, **ABL**, **Culture**, **Performance**, **Environment**, and **Common**, as well as the settings for OpenEdge AppServer launch configurations.

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.

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><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></td>
</tr>
<tr>
<td><strong>Enable logging</strong></td>
<td>If checked, makes other fields on this tab editable, allowing you to set client logging options.</td>
<td></td>
</tr>
<tr>
<td><strong>Logging level</strong></td>
<td>Specifies the level at which entries are written to the client log file.</td>
<td>Logging Level (-logginglevel)</td>
</tr>
<tr>
<td><strong>Logging file count</strong></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><strong>Logging size threshold</strong></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 | Purpose | Related parameter (if applicable)
--- | --- | ---
Append to log file | 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. | Record Buffer Size (\(-\text{Mr}\))
Clear log files | If checked, causes OpenEdge, when the session starts, to delete existing log files matching the specified client log file name. | Clear Log (\(-\text{clearlog}\))
Log file | Specifies a name and location (full path or path relative to the working directory) for the client log file. | Client Logging (\(-\text{clientlog}\))
Log entry list | 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. | 
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 OpenEdge Architect Debugger. | 
Close | Closes the Configurations wizard. | 

### Related Concepts
Launch configurations for running and debugging programs

### Related Tasks
Using launch configurations

### Related Reference
Main tab (Application)
Startup tab (Application)
PROPATH tab (Application)
Databases tab (Application)
ABL tab (Application)
Culture tab (Application)
Performance tab (Application)
Environment tab (Application)
Common tab (Application)
Environment tab (Application)

The Environment tab in the 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.

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.

Also see help for the other tabs of the Configurations wizard: Main, Startup, PROPATH, Databases, ABL, Performance, Logging, Culture, and Common, as well as the settings for OpenEdge AppServer launch configurations.

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>
<tr>
<td>New</td>
<td>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.</td>
</tr>
<tr>
<td>Select</td>
<td>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.</td>
</tr>
<tr>
<td>Edit</td>
<td>Opens a dialog that lets you modify the name/value of the environment variable currently selected in the grid.</td>
</tr>
<tr>
<td>Remove</td>
<td>Deletes the currently selected environment variable from the grid.</td>
</tr>
<tr>
<td>Append environment to native environment/Replace native environment with specified environment</td>
<td>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.</td>
</tr>
</tbody>
</table>
Running and Debugging ABL Programs

<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<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>Runs the specified startup program or opens it in the OpenEdge Architect Debugger.</td>
</tr>
<tr>
<td>Close</td>
<td>Closes the Configurations wizard.</td>
</tr>
</tbody>
</table>

**Related Concepts**
Launch configurations for running and debugging programs

**Related Tasks**
Using launch configurations

**Related Reference**
- Main tab (Application)
- PROPATH tab (Application)
- Databases tab (Application)
- Culture tab (Application)
- Performance tab (Application)
- Logging tab (Application)
- Common tab (Application)
- Launching preferences
Common tab (Application)

The Common tab in the OpenEdge Application section of the Configurations wizard lets you set options for where an ABL application launch configuration is saved, how it is accessed, and how it runs.

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.

Also see help for the other tabs of the Configurations wizard: Main, Startup, PROPATH, Databases, ABL, Culture, Performance, Logging, and Environment, as well as the settings for OpenEdge AppServer launch configurations.

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>Tells OpenEdge Architect 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>
</tbody>
</table>
| Favorites menu selector         | 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.  
**Note:** You can also manage these menu options on via Organize Favorites on the Run and Debug menus. |
<p>| Console Encoding: Default/Other | Specifies the code page used for program output to the Eclipse console, either the default or a selected non-default code page. |
| Allocate Console               | If checked, allows program output to appear in the Eclipse console. |
| File                            | If checked, redirects program output to the specified file. |
| Append                          | If checked, tells OpenEdge Architect to append redirected program output to the specified file. Otherwise, the file is overwritten. |
| Launch in background            | If checked, lets you perform other tasks while the configuration runs. |</p>
<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<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>Runs the specified startup program or opens it in the OpenEdge Architect Debugger.</td>
</tr>
<tr>
<td>Close</td>
<td>Closes the Configurations wizard.</td>
</tr>
</tbody>
</table>

**Related Concepts**
Launch configurations for running and debugging programs

**Related Tasks**
Using launch configurations

**Related Reference**
- Main tab (Application)
- Startup tab (Application)
- PROPATH tab (Application)
- Databases tab (Application)
- ABL tab (Application)
- Culture tab (Application)
- Performance tab (Application)
- Logging tab (Application)
- Environment tab (Application)
- Launching preferences
OpenEdge AppServer settings

The topics in this section describe the options and controls available for defining OpenEdge AppServer launch configurations in the Configurations wizard. There is a separate topic for each of the following tabs in this section of the wizard:

- Server
- Startup
- PROPATH
- Databases
- Common
Server tab (AppServer)

The **Server** tab in the OpenEdge AppServer section of the Configurations wizard specifies the AppServer instance used by the launch configuration.

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," you have access to all launch configurations and the options are identical regardless of how you open the wizard.

Also see help for the other tabs of the Configurations wizard: **Startup**, **PROPATH**, **Databases**, and **Common**, as well as the settings for OpenEdge Application launch configurations.

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.

The following controls are available on the **Main** tab:

<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>Server</strong></td>
<td>(Required) Specifies the AppServer 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 instance. You can change this value in the <strong>Server Editor</strong> 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 runs. OpenEdge Architect currently provides AppServer support only on the local machine.</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 the specified AppServer instance. <strong>Note:</strong> The OpenEdge Architect Debugger does not support AppServer debugging. However, the <strong>Debug</strong> button is enabled, and you must start the AppServer with the <strong>Debug</strong> command if you intend to debug it with the standalone OpenEdge Debugger.</td>
</tr>
<tr>
<td><strong>Close</strong></td>
<td>Closes the Configurations wizard.</td>
</tr>
</tbody>
</table>
Related Concepts

Launch configurations for running and debugging programs
Introducing Appserver support in OpenEdge_Architect

Related Tasks

Managing launch configurations
Starting an AppServer instance

Related Reference

Startup tab (AppServer)
PROPATH tab (AppServer)
Databases tab (AppServer)
Common tab (AppServer)
OpenEdge Application settings
Running and Debugging ABL Programs

Startup tab (AppServer)

The **Startup** tab in the OpenEdge AppServer section of the Configurations wizard specifies options for starting the AppServer instance used by the launch configuration.

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," you have access to all launch configurations and the options are identical regardless of how you open the wizard.

Also see help for the other tabs of the Configurations wizard: **Server**, **PROPATH**, **Databases**, and **Common**, as well as the settings for **OpenEdge Application launch configurations**.

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.

The following controls are available on the **Main** tab:

<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name (accessible from all tabs)</strong></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><strong>Working directory</strong></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 broker is used. Click Workspace, File System, or Variables to specify a location by browsing or using a variable.</td>
</tr>
<tr>
<td><strong>Activate</strong> <strong>Deactivate</strong> <strong>Connect</strong> <strong>Disconnect</strong> <strong>Startup</strong> <strong>Shutdown</strong></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><strong>Startup procedure parameter</strong></td>
<td>(Optional) Specifies one or more startup parameters to be used when the startup procedure, if any, runs.</td>
</tr>
<tr>
<td><strong>AppServer startup parameters</strong></td>
<td>(Optional) Specifies one or more startup parameters to be used when the server instance starts.</td>
</tr>
</tbody>
</table>
### Control

<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Import From Server</strong></td>
<td>Copies the <strong>Startup</strong> tab values from the AppServer broker stored with the AdminServer. Note that PROPATH and database information are not copied.</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 the specified AppServer instance. <strong>Note:</strong> The OpenEdge Architect Debugger does not support AppServer debugging. However, the <strong>Debug</strong> button is enabled, and you must start the AppServer with the <strong>Debug</strong> command if you intend to debug it with the standalone OpenEdge Debugger.</td>
</tr>
<tr>
<td><strong>Close</strong></td>
<td>Closes the Configurations wizard.</td>
</tr>
</tbody>
</table>

### Related Concepts
- Launch configurations for running and debugging programs
- Introducing AppServer support in OpenEdge Architect

### Related Tasks
- Managing launch configurations
- Starting an AppServer instance

### Related Reference
- Server tab (AppServer)
- PROPATH tab (AppServer)
- Databases tab (AppServer)
- Common tab (AppServer)
- OpenEdge Application settings
PROPATH tab (AppServer)

The PROPATH tab in the OpenEdge AppServer section of the Configurations wizard lets you control where OpenEdge searches for files and procedures when the launch configuration runs.

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," you have access to all launch configurations and the options are identical regardless of how you open the wizard.

Also see help for the other tabs of the Configurations wizard: Server, Startup, Databases, and Common, as well as the settings for OpenEdge Application launch configurations.

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>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>Tree view</td>
<td>Shows the current PROPATH definition.</td>
</tr>
<tr>
<td>Add Project</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>Copy ini Settings</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>Other command buttons</td>
<td>Function as described in the PROPATH help for the Project Properties dialog.</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 the specified AppServer instance.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The OpenEdge Architect Debugger does not support AppServer debugging. However, you must start the AppServer with the Debug command if you intend to debug it with the standalone OpenEdge Debugger.</td>
</tr>
<tr>
<td>Close</td>
<td>Closes the Configurations wizard.</td>
</tr>
</tbody>
</table>
Note: The only OpenEdge substitution variables that this tab supports are \texttt{@WORK} and \texttt{@DLC}. In addition, the Eclipse substitution variables (\texttt{Window>Preferences>Run/Debug>String Substitution}) are supported.

\textbf{Related Concepts}

Launch configurations for running and debugging programs  
Introducing AppServer support in OpenEdge Architect  
AppServer debugging

\textbf{Related Tasks}

Managing launch configurations  
Starting an AppServer instance

\textbf{Related Reference}

Server tab (AppServer)  
Databases tab (AppServer)  
Common tab (AppServer)  
OpenEdge Application settings
Databases tab (AppServer)

The **Databases** tab in the OpenEdge AppServer section of the Configurations wizard lets you specify the OpenEdge database connections used by an AppServer launch configuration.

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," you have access to all launch configurations and the options are identical regardless of how you open the wizard.

Also see help for the other tabs of the Configurations wizard: **Server**, **Startup**, **PROPATH**, and **Common**, as well as the settings for OpenEdge Application launch configurations.

The **Databases** tab functions like the Database Connections page in the Project Properties dialog, with the following controls 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>Show selected/Show all</strong></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><strong>Database connection list</strong></td>
<td>Lists available OpenEdge database connections, filtered according to the <strong>Show selected/Show all/Show from project</strong> 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 Database Connections help 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>Discards unsaved changes to the current launch configuration definition.</td>
</tr>
</tbody>
</table>
| **Run or Debug**                 | Starts the specified AppServer instance.  
**Note:** The OpenEdge Architect Debugger does not support AppServer debugging. However, the **Debug** button is enabled, and you must start the AppServer with the **Debug** command if you intend to debug it with the standalone OpenEdge Debugger. |
| **Close**                        | Closes the Configurations wizard.                                         |
Related Concepts
Launch configurations for running and debugging programs
Introducing AppServer support in OpenEdge Architect

Related Tasks
Managing launch configurations
Starting an AppServer instance

Related Reference
Server tab (AppServer)
Startup tab (AppServer)
PROPATH tab (AppServer)
Common tab (AppServer)
OpenEdge Application settings
Common tab (AppServer)

The **Common** tab in the OpenEdge AppServer section of the Configurations wizard lets you set options for where an AppServer launch configuration is saved, how it is accessed, and how it runs.

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," you have access to all launch configurations and the options are identical regardless of how you open the wizard.

Also see help for the other tabs of the Configurations wizard: **Server**, **Startup**, **PROPATH**, and **Databases**, as well as the settings for OpenEdge Application launch configurations.

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>Tells OpenEdge Architect 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. <strong>Note:</strong> You can also manage these menu options on via Organize Favorites on the Run and Debug menus.</td>
</tr>
<tr>
<td>Console 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, tells OpenEdge Architect 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>
</tbody>
</table>
Running and Debugging ABL Programs

<table>
<thead>
<tr>
<th>Control</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revert</td>
<td>Discards unsaved changes to the current launch configuration definition.</td>
</tr>
<tr>
<td>Run or Debug</td>
<td>Starts the specified AppServer instance.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The OpenEdge Architect Debugger does not support AppServer debugging. However, the <strong>Debug</strong> button is enabled, and you must start the AppServer with the <strong>Debug</strong> command if you intend to debug it with the standalone OpenEdge Debugger.</td>
</tr>
<tr>
<td>Close</td>
<td>Closes the Configurations wizard.</td>
</tr>
</tbody>
</table>

**Related Concepts**
- Launch configurations for running and debugging programs
- Introducing AppServer support in OpenEdge Architect

**Related Tasks**
- Managing launch configurations
- Starting an AppServer instance

**Related Reference**
- Server tab (AppServer)
- Startup tab (AppServer)
- PROPATH tab (AppServer)
- Databases tab (AppServer)
- OpenEdge Application settings
Launching preferences

The **Launching** page (**OpenEdge Architect>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:

| **Use project AVM for new launch configurations** | 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. |
| **Use selection in workspace for new launch configurations** | 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. |

**Related Concepts**
- Launch configurations for running and debugging programs

**Related Tasks**
- Managing launch configurations

**Related Reference**
- Launch configuration settings
Debugger dialogs

The topics in this section include:

- Debug Preferences dialog
- Variable Details dialog
- Select Columns dialog
- Set Value dialog
- New Breakpoint dialog
- Edit Breakpoint dialog
- Add Watch Expression dialog
- Edit Watch Expression dialog
- Dynamic Object Monitoring dialog
- Filter Objects dialog
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>OpenEdge Architect>Debug**.

You can set the following preferences:

<table>
<thead>
<tr>
<th>Preference</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;<strong>Property evaluation disabled</strong>&quot; appears. This preference is disabled by default.</td>
</tr>
<tr>
<td>Connection timeout (ms)</td>
<td>Sets the time-out used by the Debugger to connect to the OpenEdge Client. The default value is 20000 ms.</td>
</tr>
</tbody>
</table>
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>

**Related Concept**
Options for monitoring results

**Related Tasks**
Using the Variable Details dialog
Using the Variables view

**Related Reference**
Expressions view
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>
<tr>
<td>Value</td>
<td>The value of the variable.</td>
</tr>
<tr>
<td>Actual Type</td>
<td>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.</td>
</tr>
</tbody>
</table>

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.)

Related Task

Using the Variables view

Related Reference

Variables view
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.

*Related Task*

- Changing program code

*Related Reference*

- Variables view
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**.

**Related Concept**

- Breakpoints

**Related Task**

- Setting breakpoints

**Related Reference**

- Breakpoints view
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.

Related Concept
Breakpoints

Related Task
Setting breakpoints

Related Reference
Breakpoints view
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.

**Related Concept**
- Options for monitoring results

**Related Task**
- Using watch expressions

**Related Reference**
- Expressions view
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.

**Related Concept**
- Options for monitoring results

**Related Task**
- Using watch expressions

**Related Reference**
- Expressions view
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>Object type tree</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</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</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</strong></td>
<td>To dismiss the dialog.</td>
</tr>
</tbody>
</table>

**Related Concept**
- Options for monitoring results

**Related Task**
- Using the Dynamic Objects view

**Related Reference**
- Dynamic Objects view
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.

Related Concept
Options for monitoring results

Related Task
Using the Dynamic Objects view

Related Reference
Dynamic Objects view
Debugger views

The topics in this section include:

- Debug view
- Variables view
- Breakpoints view
- Expressions view
- Dynamic Objects view
Running and Debugging ABL Programs

Debug view

The Debug view shows a session tree for each active Debugger session. It also provides the program execution commands that you use to step through the code.

The OpenEdge Debugger 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

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>Step Into</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 Code stepping.</td>
</tr>
</tbody>
</table>
### Running and Debugging ABL Programs

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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.</td>
</tr>
<tr>
<td><strong>Terminate</strong></td>
<td>Interrupt execution and end the Debugger session.</td>
</tr>
<tr>
<td><strong>Remove All Terminated Launches</strong></td>
<td>Clear the <strong>Debug</strong> view of session trees that are no longer active.</td>
</tr>
</tbody>
</table>

**Related Concepts**
- The OpenEdge Debugger perspective
- Code-stepping
- Options for monitoring results

**Related Tasks**
- Using the Debug view and stack trace
- Stepping through the code
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 OpenEdge Architect 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 OpenEdge Debugger perspective includes the **Variables** view by default. To display it if it is not open, select **Window> Show View> Variables**.

The following commands are available:

<table>
<thead>
<tr>
<th><strong>Collapse All</strong></th>
<th>To collapse the display of all hierarchical data elements in the list, leaving only the top-level entities visible.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>To open a drop-down menu that provides options for customizing the appearance of the view. The <strong>Filter</strong> option lets you select the data element types that you want to include in the <strong>Variables</strong> view.</td>
</tr>
</tbody>
</table>
The drop-down menu provides these options:

<table>
<thead>
<tr>
<th><strong>Layout</strong></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. |

**Related Concepts**
- Options for monitoring results
- The OpenEdge Debugger perspective

**Related Tasks**
- Using the Variables view
- Using the Variable Details dialog

**Related Reference**
- Variable Details dialog
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. You can selectively enable or disable breakpoints for the current debugging session.

The OpenEdge Debugger perspective includes the **Breakpoints** view by default. To display it if it is not open, select **Window>Show View>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:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Remove Selected Breakpoints</strong></td>
<td>To permanently delete currently selected breakpoints.</td>
</tr>
<tr>
<td><strong>Remove All Breakpoints</strong></td>
<td>To permanently delete all breakpoints defined for the workspace.</td>
</tr>
<tr>
<td><strong>Skip All Breakpoints</strong></td>
<td>To disable all breakpoints without deleting them.</td>
</tr>
</tbody>
</table>

The following commands are available only if the breakpoints are grouped:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Expand All</strong></td>
<td>To expand all the items in the view.</td>
</tr>
<tr>
<td><strong>Collapse All</strong></td>
<td>To collapse all the items in the view.</td>
</tr>
</tbody>
</table>

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>

**Related Concepts**

- Options for controlling program execution
- Options for monitoring results
- The OpenEdge Debugger perspective

**Related Task**

- Using breakpoints
Related Reference

- New Breakpoint dialog
- Edit Breakpoint dialog
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 OpenEdge Debugger perspective includes the **Expressions** view by default. To display it if it is not open, select **Window > Show View > Expressions**.

Each entry label shows the expression and its current value, "Unavailable," or "(disabled)."

The following commands are available:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Collapse All</strong></td>
<td>To collapse the display of all hierarchical data elements in the list, leaving only the top-level entities visible.</td>
</tr>
<tr>
<td><strong>Remove</strong></td>
<td>To delete currently selected watch expressions.</td>
</tr>
<tr>
<td><strong>Remove All</strong></td>
<td>To delete all watch expressions defined for the workspace.</td>
</tr>
</tbody>
</table>

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>

**Related Concepts**

Options for monitoring results  
The OpenEdge Debugger perspective

**Related Tasks**

Using watch expressions  
Setting breakpoints  
Using the Variables view  
Using the Variable Details dialog
Running and Debugging ABL Programs

**Related Reference**

- Add Watch Expression dialog
- Edit Watch Expression dialog
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 OpenEdge Debugger 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 > 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.

**Related Concepts**

- Options for monitoring results
- The OpenEdge Debugger perspective

**Related Task**

- Using the Dynamic Objects view

**Related Reference**

- Dynamic Object Monitoring dialog
- Filter Objects dialog
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. Progress 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.

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 Progress DB Navigator consists of the following sections:

**Concepts**
- Progress DB Navigator perspective
- SQL Connection profiles
- JDBC drivers

**Tasks**
- Common tasks for supported databases
- Working with MS SQL Server databases
- Working with OpenEdge databases
- Working with Oracle databases

**Reference**
- Components of the Progress DB Navigator perspective
- MS SQL Server database
- OpenEdge database
- Oracle database
- Progress DB Navigator preferences
Concepts

This section includes the following topics:

- Progress DB Navigator perspective
- SQL Connection profiles
- JDBC drivers
Progress DB Navigator perspective

In Eclipse, perspectives define the layout of the initial set of views in the Workbench window. Progress DB Navigator includes the Progress 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 Progress DB Navigator 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 Progress DB Navigator perspective also provides access to the SQL Editor, which allows you to write and execute SQL scripts.

**Related Concept**

SQL Connection profiles

**Related Task**

Accessing the Progress DB Navigator perspective

**Related Reference**

Components of the Progress DB Navigator perspective
SQL Connection profiles

Progress 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 DB Navigator > Connection Profiles** page.

**Related Task**

Adding SQL connection profiles

**Related Reference**

SQL Connection Profile preferences
JDBC drivers

JDBC (Java Database Connectivity) drivers allow Java-based applications to connect to SQL-compliant databases.

Progress 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 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).

Related Concept
SQL connection profiles

Related Tasks
Configuring a JDBC driver
Connecting to a database
Context-sensitive help

This section includes the following topics:

- Common tasks for supported databases
- Working with MS SQL Server databases
- Working with OpenEdge databases
- Working with Oracle databases
Common tasks for supported databases

The following tasks are commonly performed when you use Progress DB Navigator to work with a SQL-compliant database:

- Setting database preferences
- Accessing the Progress DB Navigator perspective
- Configuring a JDBC driver
- Connecting to a database
- Working with SQL connection profiles
- SQL scripting
- Working with application data

Related Tasks

Adding a SQL connection profile for an MS SQL Server database
Adding a SQL connection profile for an OpenEdge database
Adding a SQL connection profile for an Oracle database
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.

**Related Tasks**
- Connecting to a database
- Adding SQL connection profiles
- Configuring a JDBC driver

**Related Reference**
- SQL Editor
Accessing the Progress DB Navigator perspective

You can open the Progress DB Navigator perspective in different ways. For example, you can:

1. Click **Open Perspective** on the Workbench toolbar.
2. Select **Progress DB Navigator** from the drop-down menu if it is listed. If it is not listed, choose **Other**. Then choose **Progress DB Navigator** from the **Open Perspective** dialog.

You can also:

1. Select ***Window***>**Open Perspective**.
2. Select **Progress DB Navigator** if it is listed, or select **Other** and then choose **Progress DB Navigator** from the **Open Perspective** dialog.

If you have already opened the Progress DB Navigator in a workspace, you can access it again by choosing **Progress 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.

After you open the Progress DB Navigator perspective, the next step is to connect to a database. See **Connecting to a database** for more information.

**Related Reference**

For more information on using perspectives, toolbars, and menu bars, see the Workbench User Guide in the Eclipse help system.
Configuring a JDBC driver

Progress 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 Progress 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, [thehostmachine]. (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.

Troubleshooting the JDBC drivers

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
**Related Concept**

JDBC drivers

**Related Task**

Connecting to a database
Connecting to a database

You can connect Progress DB Navigator to multiple databases.

The following is an overview of how to connect Progress DB Navigator to a database (follow the links for more information about the tasks):

1. 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 Progress 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.

2. Configure a JDBC driver for the database.

   Progress 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 Progress DB Navigator. If, however, the JDBC driver for your database requires driver configuration, see Troubleshooting the JDBC drivers for more information.

3. 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 Progress 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, note that ABL access is required for trigger maintenance. Therefore, create a server with a client type of **Both 4GL and SQL** (which is the default).

4. 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.

   When you open a connection profile for a database, the schema for the database appears in the **DB Structure** view.

**Related Concepts**

- SQL connection profiles
- JDBC drivers
Related Tasks

- Configuring a JDBC driver
- Adding SQL connection profiles
- Opening and closing SQL connections

Related Reference

- Connections view
- Connection Info view
- DB Structure view
Working with SQL connection profiles

Working with SQL connection profiles in Progress DB Navigator includes the following tasks:

- Adding SQL connection profiles
- Editing SQL connection profiles
- Displaying SQL connection profiles
- Opening and closing SQL connections

Related Concept
SQL Connection profiles
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 the Progress DB Navigator.

Progress 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 Progress DB Navigator. You may, however, find that you need to configure the driver, and you can do so from the Window>Preferences>Progress 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.

Related Concept

SQL connection profiles

Related Tasks

Configuring a JDBC driver
Connecting to a database

Related Reference

SQL Connection profile preferences
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: Click Test Connection to verify database connectivity.

Related Concept
  SQL connection profiles

Related Task
  Adding SQL connection profiles

Related Reference
  SQL Connection Profile preferences
Displaying SQL connection profiles

You can access a list of SQL connection profiles by using either of the following methods:

- Select **Window > Preferences** from the main menu bar, and then select the **Progress DB Navigator** node from the preference tree. Select **Connection Profiles** in the expanded tree view.
- From the **Connections view**, expand the **Connection Profiles** node.

**Related Concept**
SQL connection profiles

**Related Tasks**
- Adding SQL connection profiles
- Editing SQL connection profiles
- Opening and closing SQL connections

**Related Reference**
SQL Connection Profile preferences
Opening and closing SQL connections

To open (start) a connection:

1. Select **Window> Show View> Connections** to access the **Connections view** in Progress DB Navigator.

2. Double-click a connection profile beneath the **Connection Profiles** node. (Or, select a connection profile and click **Open Connection**.) The **Connection** dialog appears.

   **Note:** If you selected the **Auto-Login on connect** option when you created the connection profile, the system attempts to connect to the database, displaying the schema in the **DB Structure view**. Otherwise, the **Connection** dialog appears.

3. 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.

   **Note:** If you intend to view data in or modify an OpenEdge database, the user account must have DBA privileges.

4. Choose either **AutoCommit** or **Commit On Close** in the **Connections** view tool bar.

   **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.

5. Click **OK**.

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.

To close a connection:

1. Access the **Connections view** in Progress DB Navigator.

2. Select the connection profile of the connection you want to close, located under the **Active Connections** node.

3. Click **Close Connection**. (Alternatively, click **Close All Connections**.)

**Related Concepts**

- SQL connection profiles
- JDBC Drivers

**Related Tasks**

- Configuring a JDBC driver
- Adding SQL connection profiles
Related Reference

Connections view
Connection dialog
Progress 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.

SQL scripting includes the following tasks:

- Starting the SQL editor
- Generating SQL queries
- Copying and pasting
- Generating a create table script
- Saving SQL scripts
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 ProgressDB 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.

**Related Tasks**

Copying and pasting
Generating SQL queries
Generating a create table script
Saving SQL scripts
Related Reference

SQL Editor
SQL Editor preferences
SQL History view
SQL Results view
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 browse 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.

Related Tasks

- Starting the SQL editor
- Copying and pasting
- Generating a create table script
- Saving SQL scripts

Related Reference

- SQL Editor
- SQL Editor Preferences
- SQL History view
- SQL Results view
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 **Progress DB Navigator preferences** 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.)

**Related Tasks**

- Starting the SQL editor
- Generating SQL queries
- Generating a create table script
- Saving SQL scripts

**Related Reference**

- SQL Editor
- SQL Editor preferences
- SQL History view
- SQL Results view
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.

**Notes:** 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.

**Related Tasks**
- Starting the SQL editor
- Copying and pasting
- Generating SQL queries
- Saving SQL scripts

**Related Reference**
- SQL Editor
- SQL Editor preferences
- SQL History view
- SQL Results view
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.

Related Tasks
Starting the SQL editor
Copying and pasting
Generating a SQL query
Generating a create table script

Related Reference
SQL Editor
SQL Editor preferences
SQL History view
SQL Results view
Working with application data

To view application schema using Progress 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.

Working with application data includes the following tasks:

- **Viewing application data**
- **Editing application data**
Viewing application data

From the Progress 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.

To preview a limited set of records:

1. Select a table name in the DB Structure view.
2. Select the Preview tab in the DB Details view.

   **Note:** By setting the Preview row limit in the Progress DB Navigator preferences, you can adjust the number of rows displayed in the Preview tab.

To view all data in a table:

1. Select a table name in the DB Structure view.
2. Right-click on the table and choose Generate Select in SQL Editor to open a SQL editor containing the generated SELECT statement.
3. 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 Progress DB Navigator preferences.

You can also filter the rows displayed by entering a SQL WHERE statement. For example, the following statement displays only those rows in which the value in the Balance column is greater than 10000:

```
WRITE your where condition: Balance > 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 Progress 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 Progress DB Navigator allows you to:

- Set preferences
- Add a SQL connection profile for the database
- Troubleshoot the MS SQL Server JDBC driver
- View 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.

**Related Tasks**
- Connecting to a database
- Adding SQL connection profiles
- Configuring a JDBC driver

**Related Reference**
- SQL Editor
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 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.
   
   (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 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**. If the connection succeeds, a **Connection succeeded** message appears. If the connection does not succeed, an error message appears.
13. Click **OK**.

**Related Tasks**

Troubleshooting the MS SQL Server JDBC driver
Troubleshooting the MS SQL Server JDBC driver

Progress 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 Progress 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.
   - For the Microsoft SQL Server JDBC Driver (DataDirect), the URL should look similar to the following:
     \[jdbc:progress:sqlserver://hostname:port;DatabaseName=name\]
     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, \([\text{thehostmachine}]\).
4. Close the Edit Connection Profile dialog once you confirm that the URL is correct.
5. Choose Window>Preferences>Progress 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:
      \[\text{com.prgs.jdbc.sqlserver.SQLServerDriver}\]
   c. Click the Extra Class Path tab. Verify that the following pathnames appear:
      \[
      \text{install_dir}\text{\oeide\architect\eclipse\plugins\com.progress.dbnavigator_2.1.0\drivers\pgsqlserver.jar}
      \text{install_dir}\text{\oeide\architect\eclipse\plugins\com.progress.dbnavigator_2.1.0\drivers\pgbase.jar}
      \text{install_dir}\text{\oeide\architect\eclipse\plugins\com.progress.dbnavigator_2.1.0\drivers\pgutil.jar}
      \]
The `install_dir` path refers to the complete path of your product installation. You must provide this complete pathname to configure the driver properly.

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.

**Related Concept**
- JDBC drivers

**Related Task**
- Connecting to a database
Viewing MS SQL Server database schema

You can view MS SQL server database schema as follows:

- Displaying MS SQL Server database function details
- Displaying MS SQL Server database procedure details
- Displaying MS SQL Server database session details
- Displaying MS SQL Server database table details
- Displaying MS SQL Server database triggers
- Displaying MS SQL Server database view details

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.

**Related Reference**
- MS SQL Server DB Structure view
- MS SQL Server DB Details view
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.

**Related Reference**
- MS SQL Server DB Structure view
- MS SQL Server DB Details view
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.

Related Reference

MS SQL Server DB Structure view
MS SQL Server DB Details view
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**.

**Related Reference**
- MS SQL Server DB Structure view
- MS SQL Server DB Details view
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.

**Related Reference**

- MS SQL Server DB Structure view
- MS SQL Server DB Details view
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.

**Related Reference**
- MS SQL Server DB Structure view
- MS SQL Server DB Details view
Working with OpenEdge databases

Working with an OpenEdge database using Progress DB Navigator allows you to:

- Set preferences
- Add a SQL connection to the database
- Troubleshoot the Progress OpenEdge JDBC driver
- Create a new database
- Work with user accounts
- View and maintain 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.

**Related Tasks**

- Connecting to a database
- Adding SQL connection profiles
- Configuring a JDBC driver

**Related Reference**

- SQL Editor
Adding a SQL connection profile for an OpenEdge 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 **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.

   (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 that 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**.

   If you are connecting to an OpenEdge Personal database (which is the default RD-BMS packaged with OpenEdge Architect), 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.

8. 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 Progress Explorer.

9. Type the logical name of the database in the **Database** field. Check for the correct name in the **Databases** folder in Progress Explorer.

10. Add any additional JDBC connection properties in the **Additional parameters** field, delimited by semicolons in the format `key1=value; key2=value`.

11. 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.

12. Click **OK**.

**Related Concept**

**SQL Connection profiles**

**Related Task**

**Working with SQL Connection profiles**

**Related Reference**

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.
Troubleshooting the Progress OpenEdge JDBC driver

Progress 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 Progress 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.

   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 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 as follows:

      `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.jar`
      `install_dir\oeide\architect\eclipse\plugins\com.progress.dbnavigator_2.1.0\drivers\base.jar`
      `install_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.

**Related Concept**

- JDBC drivers

**Related Task**

- Connecting to a database
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.

To connect to the database after creating it, you must create a SQL connection profile and start a server for the database.

Related Concept
SQL connection profiles

Related Tasks

- Adding SQL connection profiles
- Adding a SQL connection profile for an OpenEdge database

Related Reference
Connections view
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.

Related Reference

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.
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 Progress 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.

**Caution:** 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.

Working with user accounts includes the following tasks:

- Creating OpenEdge database user accounts
- Viewing OpenEdge database user account information
- Changing OpenEdge database user account passwords
- Deleting OpenEdge database user accounts
- Granting privileges to OpenEdge database user accounts
- Changing privileges to OpenEdge database user accounts
Creating OpenEdge database user accounts

To create 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 becomes active in the **DB Details view**.
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 into Progress 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.
   
   **Caution**: The password is case-sensitive; user IDs are not case-sensitive.
6. Choose to grant either DBA or RESOURCE privileges for the user.
   
   **Caution**: 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.

**Related Tasks**

- Viewing OpenEdge database user account information
- Changing OpenEdge database passwords
- Deleting OpenEdge database user accounts
- Granting OpenEdge database privileges
Changing OpenEdge database privileges
Connecting to a database
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><strong>User ID</strong></th>
<th>Either the user ID defined in the _User table or the user SQL authorization.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></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 the Progress DB Navigator.</td>
</tr>
<tr>
<td><strong>Account Type</strong></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><strong>DBA Rights</strong></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. Only users with DBA privileges can use the Edit Schema wizards for maintaining the database.</td>
</tr>
<tr>
<td><strong>Resource Rights</strong></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>

**Related Tasks**

- Creating OpenEdge database user accounts
- Changing OpenEdge database passwords
- Deleting OpenEdge database user accounts
- Changing OpenEdge database privileges
- Granting OpenEdge database privileges

**Related Reference**

See OpenEdge Data Management: SQL Development on the PSDN Product Documentation Web site for more information on managing SQL user accounts.
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.

Related Tasks

Viewing OpenEdge database user account information
Creating OpenEdge database user accounts
Deleting OpenEdge database user accounts
Changing OpenEdge database privileges
Granting OpenEdge database privileges
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.

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.

Related Tasks

- Viewing OpenEdge database user account information
- Creating OpenEdge database user accounts
- Changing OpenEdge database passwords
- Granting OpenEdge database privileges
- Changing OpenEdge database privileges
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.
2. Choose the Security node in the DB Structure view.
   
   The Security tab appears in the DB Detail view.
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.

Related Tasks

- Viewing OpenEdge database user account information
- Creating OpenEdge database user accounts
- Changing OpenEdge database passwords
- Deleting OpenEdge database user accounts
- Changing OpenEdge database privileges
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.

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.

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).

**Related Tasks**
- Viewing OpenEdge database user account information
- Creating OpenEdge database user accounts
- Changing OpenEdge database passwords
- Deleting OpenEdge database user accounts
- Granting OpenEdge database privileges
Viewing and maintaining OpenEdge database schema

After you connect to an OpenEdge database, Progress 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 Progress DB Navigator perspective.

Viewing and maintaining OpenEdge database schema includes working with the following:

- OpenEdge database tables
- OpenEdge database columns
- OpenEdge database indexes
- OpenEdge database sequences
- OpenEdge database triggers
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.

Working with OpenEdge database tables includes the following tasks:

- Displaying OpenEdge database table details
- Adding OpenEdge database tables
- Editing OpenEdge database table properties
- Dropping OpenEdge database tables
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.

**Related Tasks**
- Editing OpenEdge database table properties
- Adding OpenEdge database tables
- Dropping OpenEdge database tables
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. In the Area field, select a storage area from the list.

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 Add Table wizard for more information about these fields.

6. 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.

7. 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.

**Related Tasks**

- Displaying OpenEdge database table details
- Editing OpenEdge database table properties
- Dropping OpenEdge database tables

**Related Reference**

- OpenEdge table wizard
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**.

**Related Tasks**
- Displaying OpenEdge database table details
- Adding OpenEdge database tables
- Dropping OpenEdge database tables

**Related Reference**
- OpenEdge Edit Schema wizard
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.

**Related Tasks**

- Displaying OpenEdge database table details
- Editing OpenEdge database table properties
- Adding OpenEdge database tables
Working with OpenEdge database table columns

In the Progress DB Navigator perspective, 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.

Working with OpenEdge database table columns includes the following tasks:

- Displaying OpenEdge database column details
- Adding OpenEdge database columns
- Editing OpenEdge database column details
- Copying OpenEdge database columns
- Dropping OpenEdge database columns

**Related Reference**

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 **OpenEdgeColumns** 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.

**Related Tasks**
- Adding OpenEdge database columns
- Editing OpenEdge database column properties
- Copying OpenEdge database columns
- Dropping OpenEdge database columns
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**.

Columns appear in the DB Structure view by creation date rather than by Order property value.

**Related Tasks**

- Displaying OpenEdge database column details
- Editing OpenEdge database column details
- Copying OpenEdge database columns
- Dropping OpenEdge database columns

**Related Reference**

- OpenEdge Columns wizard
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 Progress DB Navigator.

4. Click Finish.

**Related Tasks**

- Displaying OpenEdge database column details
- Adding OpenEdge database columns
- Copying OpenEdge database columns
- Dropping OpenEdge database columns

**Related Reference**

- OpenEdge Columns wizard
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.

**Related Tasks**

- Displaying OpenEdge database column details
- Editing OpenEdge database column details
- Adding OpenEdge database columns
- Dropping OpenEdge database columns

**Related Reference**

- OpenEdge Columns wizard
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.

**Related Tasks**

- Displaying OpenEdge database column details
- Editing OpenEdge database column details
- Adding OpenEdge database columns
- Copying OpenEdge database columns
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 Progress DB Navigator perspective, 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.

Working with OpenEdge database indexes includes the following tasks:

- Displaying OpenEdge database index details
- Adding OpenEdge database indexes
- Editing OpenEdge database index properties
- Dropping OpenEdge database indexes
- Enabling OpenEdge database large index keys
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.

Related Tasks

- Editing OpenEdge database index properties
- Adding OpenEdge database indexes
- Dropping OpenEdge database indexes
- Enabling OpenEdge database large index keys

Related Reference

- Index wizard
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.

Related Tasks

- Displaying OpenEdge database index details
- Editing OpenEdge database index properties
- Dropping OpenEdge database indexes
- Enabling OpenEdge database large index keys

Related Reference

Index wizard
**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**.

**Related Tasks**

- Displaying OpenEdge database index details
- Adding OpenEdge database indexes
- Dropping OpenEdge database indexes
- Enabling OpenEdge database large index keys

**Related Reference**

- Index wizard
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.

**Related Tasks**
- Displaying OpenEdge database index details
- Editing OpenEdge database index properties
- Adding OpenEdge database indexes
- Enabling OpenEdge database large index keys

**Related Reference**
- Index wizard
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.

**Note:** 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.

**Caution:** Once you enable large index keys for an index, you cannot disable them.

**Related Tasks**
- Displaying OpenEdge database index details
- Editing OpenEdge database index properties
- Adding OpenEdge database indexes
- Dropping OpenEdge database indexes

**Related Reference**
- Index wizard
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.

Working with OpenEdge database sequence includes the following tasks:

- Displaying OpenEdge database sequence properties
- Adding OpenEdge database sequences
- Editing OpenEdge database sequences
- Dropping OpenEdge database sequences
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.

**Related Tasks**
- Adding OpenEdge database sequences
- Editing OpenEdge database sequences
- Dropping OpenEdge database sequences
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. Click **Add**.
5. Click **Finish**.

The sequence is automatically added to the **DB Structure** view.

**Related Tasks**

- Displaying OpenEdge database sequence properties
- Editing OpenEdge database sequences
- Dropping OpenEdge database sequences

**Related Reference**

- Add/Edit Sequence wizard
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.

Related Tasks
- Displaying OpenEdge database sequence properties
- Adding OpenEdge database sequences
- Dropping OpenEdge database sequences

Related Reference
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.

**Related Tasks**
- Displaying OpenEdge database sequence properties
- Adding OpenEdge database sequences
- Editing OpenEdge database sequences
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 Progress DB Navigator, the DB Structure view displays three types of triggers under the TRIGGERS node in the OpenEdge schema:

<table>
<thead>
<tr>
<th>Trigger Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4GL TRIGGERS</td>
<td>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.</td>
</tr>
<tr>
<td>REPLICATION TRIGGERS</td>
<td>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 Progress DB Navigator just like other ABL triggers.</td>
</tr>
<tr>
<td>Java triggers</td>
<td>The JAVA TRIGGERS node lists all the tables that contain INSERT, DELETE, or UPDATE Java triggers.</td>
</tr>
</tbody>
</table>

**Note:** Progress DB Navigator allows you to view all three types of triggers for OpenEdge databases. However, you cannot use the Progress DB Navigator wizards to create, modify, or delete Java triggers.

**Related Tasks**
- Displaying OpenEdge database triggers
- Adding OpenEdge database triggers
- Editing OpenEdge database triggers
- Dropping OpenEdge database triggers

**Related Reference**
For more information about triggers, see the ABL manuals in the OpenEdge Product Documentation section of the Progress Communities Web site.s
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.

Related Tasks
- Adding OpenEdge database triggers
- Editing OpenEdge database triggers
- Dropping OpenEdge database triggers

Related Reference
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 website.
Creating OpenEdge database triggers

The Create Trigger wizard allows you to create a new trigger.

Note: You cannot create Java triggers using Progress 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 OpenEdge Architect ABL Editor.

Related Tasks
- Displaying OpenEdge database triggers
- Editing OpenEdge database triggers
- Dropping OpenEdge database triggers

Related Reference
- OpenEdge DB Structure view
- Create/ModifyTrigger wizard

For more information about triggers, see the ABL manuals in the OpenEdge Product Documentation section of the Progress Communities Web site.
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.

**Editing trigger code in the ABL Editor**

You can open an existing trigger and edit the code (using the OpenEdge Architect ABL Editor) by performing the following steps:

1. 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.

2. Right-click, and choose **Open in Editor** from the context menu. This option opens the trigger in the ABL Editor perspective.

   **Notes:** 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.

3. Edit and save the file.

**Editing trigger properties with the Edit Trigger option**

You can modify trigger properties by selecting a trigger name in the **DB Structure** view and choosing **Edit Trigger** from the context menu. The **Edit Trigger** dialog opens and you can:

- Choose whether to override triggers at run time.
- Choose whether to validate triggers at run time.
- 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**.

**Related Tasks**

- Displaying OpenEdge database triggers
- Adding OpenEdge database triggers
- Dropping OpenEdge database triggers
**Related Reference**

Create/ModifyTrigger wizard  
DB Structure view

For more information about triggers, see the ABL manuals in the OpenEdge Product Documentation section of the Progress Communities Web site.
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.

**Related Tasks**
- Displaying OpenEdge database triggers
- Adding OpenEdge database triggers
- Editing OpenEdge database triggers

**Related Reference**
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 website.
Working with Oracle databases

Working with an Oracle database using Progress DB Navigator allows you to:

- Set database preferences
- Add a SQL connection profile for the database
- Troubleshoot the Oracle JDBC driver
- View 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.

**Related Tasks**
- Connecting to a database
- Adding SQL connection profiles
- Configuring a JDBC driver

**Related Reference**
- SQL Editor
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.

   (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 that 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

Progress 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 Progress 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.

   For the Oracle JDBC Driver (DataDirect), the URL should look similar to the following:

   `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 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 as follows:

      `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.

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.

**Related Concept**

JDBC drivers

**Related Task**

Connecting to a database
Viewing Oracle database schema

You can view Oracle database schema as follows:

- Displaying Oracle database function details
- Displaying Oracle database package body details
- Displaying Oracle database package details
- Displaying Oracle database procedure details
- Displaying Oracle database sequence properties
- Displaying Oracle database table details
- Displaying Oracle database triggers
- Displaying Oracle database view details

Related Reference

Oracle DB Structure view
Oracle DB Details view
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.

**Related Reference**
- Oracle DB Structure view
- Oracle DB Details view
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**.

**Related Reference**

- Oracle DB Structure view
- Oracle DB Details view
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.

**Related Reference**
- Oracle DB Structure view
- Oracle DB Details view
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.

**Related Reference**

- Oracle DB Structure view
- Oracle DB Details view
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.

**Related Reference**

- Oracle DB Structure view
- Oracle DB Details view
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.

**Related Reference**

- Oracle DB Structure view
- Oracle DB Details view
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**.

**Related Reference**

- Oracle DB Structure view
- Oracle DB Details view
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**.

**Related Reference**

- Oracle DB Structure view
- Oracle DB Details view
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**.

**Related Reference**

- Oracle DB Structure view
- Oracle DB Details view
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.

**Related Reference**

- Oracle DB Structure view
- Oracle DB Details view
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.

**Related Reference**

Oracle DB Structure view  
Oracle DB Details view
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.

Related Reference

Oracle DB Structure view
Oracle DB Details view
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**.

**Related Reference**

Oracle DB Structure view
Oracle DB Details view
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**.

**Related Reference**
- Oracle DB Structure view
- Oracle DB Details view
Reference

This section includes the following topics:

- Components of the Progress DB Navigator perspective
- MS SQL Server database
- OpenEdge database
- Oracle database
- Preferences
Components of the Progress DB Navigator perspective

The topics in this section describe the views, wizards, and dialogs associated with the default Progress DB Navigator perspective, including:

- **Connection** dialog
- **Connection Info** view
- **Connections** view
- **DB Structure** view
- **DB Details** view
- **SQL Editor**
- **SQL Results** view
- **SQL History** view
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 Progress 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><strong>AutoCommit</strong></th>
<th>Any changes to the database are committed immediately. This mode is selected by default.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Commit On Close</strong></td>
<td>Any changes to the database are committed when you either close the session or select <strong>Commit</strong>.</td>
</tr>
</tbody>
</table>

**Notes**: 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.

**Related Reference**

**Connections view**
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>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Add Connection Profile" /></td>
<td>Add Connection Profile displays the Add Connection Profile dialog.</td>
</tr>
<tr>
<td><img src="image" alt="Open Connection" /></td>
<td>Open Connection opens the Connection dialog for the connection profile selected in the tree view.</td>
</tr>
<tr>
<td><img src="image" alt="New SQL Editor" /></td>
<td>New SQL Editor opens the SQL Editor.</td>
</tr>
<tr>
<td><img src="image" alt="Close All Connections" /></td>
<td>Close All Connections closes all open connections and disables any views that have related information.</td>
</tr>
<tr>
<td><img src="image" alt="Close Connection" /></td>
<td>Close Connection closes the connection that is selected in the Connections view.</td>
</tr>
<tr>
<td><img src="image" alt="Rollback" /></td>
<td>Rollback 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="image" alt="Commit" /></td>
<td>Commit 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="image" alt="Collapses" /></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="image" alt="Create OpenEdge database" /></td>
<td>Create OpenEdge database opens the Create database dialog.</td>
</tr>
</tbody>
</table>

Related Task
Connecting to a database

Related Reference
Connection dialog
Connection Info view
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 Progress DB Navigator to display or hide schemas. By default, catalogs are displayed while schemas are hidden from view. See Progress DB Navigator Preferences for more information.

When you select certain nodes in the DB Structure view, properties and values appear in the DB Details view.

Context Menus

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.

Filters

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.

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 Progress DB Navigator preferences, all catalogs appear below the database node. If you have selected Show schema in the Progress 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 sort 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.

---

**Additional nodes**

Depending on which SQL-compliant database’s structure you are viewing, the remaining nodes shown might vary.

**Related Tasks**

SQL scripting

**Related Reference**

- Progress DB Navigator preferences
- MS SQL Server DB Structure view
- OpenEdge DB Structure view
- Oracle DB Structure view
- DB Details view
- Connections view
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>Tab Folder</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Columns</strong></td>
<td>Displays a list of all columns with related SQL information.</td>
</tr>
<tr>
<td><strong>Indexes</strong></td>
<td>Displays a list of indexes in the selected table.</td>
</tr>
<tr>
<td><strong>Primary Key</strong></td>
<td>Displays the primary keys defined for the selected table. This is relevant only to tables created through SQL, although it displays the components of the OpenEdge primary key.</td>
</tr>
<tr>
<td><strong>Foreign Key</strong></td>
<td>Displays any foreign key restraints defined for a selected SQL table.</td>
</tr>
<tr>
<td><strong>Preview</strong></td>
<td>Displays a subset of the records in the selected table. In the Preferences, you can set the number of rows returned.</td>
</tr>
<tr>
<td><strong>Row Count</strong></td>
<td>Displays the total number of rows in the selected table. <strong>Note:</strong> 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.</td>
</tr>
<tr>
<td><strong>Privileges</strong></td>
<td>Displays both table and column privileges (as a toggle). Includes the columns <strong>Table</strong>, <strong>Column</strong>, <strong>Privilege</strong>, <strong>Grantor</strong>, <strong>Grantee</strong>, and <strong>Grantable</strong>.</td>
</tr>
</tbody>
</table>

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.

**Related Reference**

- DB Structure view
- MS SQL Server DB Details view
- OpenEdge DB Details view
- Oracle DB Details view
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>Action</th>
</tr>
</thead>
</table>
| ![Execute SQL](image) | Execute the SQL statement(s).  
**Note:** The Execute icon is disabled until you select a database. |
| ![Open SQL File](image) | Select and open an existing `.sql` file. |
| ![Clear](image) | Clear all content from the editor. |
| ![Database](image) | Select a database connection. When you are creating a new script, the default is the connection that is selected in the Connections view.  
**Note:** If more than one catalog exists, another drop-down list appears in which you specify which catalog to use. |

**Column Auto-Completion Assistance**

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.

**Related Tasks**

- Starting the SQL editor
- Copying and pasting
- Generating a SQL query
- Generating a create table script
- Saving SQL scripts
- Setting database preferences

**Related Reference**

- SQL Editor Preferences
SQL History view
SQL Results view
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="image" alt="Next" /></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 <strong>SQL Results rows to batch</strong> value in the Progress DB Navigator preferences page.</td>
</tr>
<tr>
<td><img src="image" alt="All" /></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="image" alt="Copy" /></td>
<td>Copy all rows in the view to the clipboard.</td>
</tr>
<tr>
<td><img src="image" alt="Close" /></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.

**Related Task**

Starting the SQL editor

**Related Reference**

SQL Editor
SQL History view
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>Action</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>

**Related Task**

- Starting the SQL editor

**Related Reference**

- SQL Editor
- SQL Results view
MS SQL Server database

You can view MS SQL Server database schema by using the following views:

- MS SQL Server DB Details view
- MS SQL Server DB 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>- <strong>Columns</strong> - A list of all columns and related SQL information. You can sort the column names in ascending or descending order by clicking the <strong>Column</strong> label.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Indexes</strong> - A list of indexes in the selected table.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Primary key</strong> - A list of the primary keys defined for the selected table.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Foreign key</strong> - A list of any foreign key restraints defined for a selected SQL table.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Preview</strong> - 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>- <strong>Row count</strong> - The total number of rows in the selected table.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Privileges</strong> - A list of both table and column privileges (as a toggle).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PROCEDURE or TRIGGER</th>
<th>For a child node, displays the SQL code in the <strong>Source</strong> tab and property/value details, such as date created, in the <strong>Info</strong> tab.</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUNCTIONS</td>
<td>Displays the SQL code in the <strong>Source</strong> tab and property/value details, such as date created, in the <strong>Info</strong> tab.</td>
</tr>
<tr>
<td>SESSIONS</td>
<td>Displays details about the current schema.</td>
</tr>
</tbody>
</table>
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 Progress DB Navigator to display or hide schemas. By default, catalogs are displayed while schemas are hidden from view. See Progress DB Navigator preferences 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 Progress DB Navigator preferences, all catalogs appear below the database node. If you have selected Show schema in the Progress 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.</td>
</tr>
<tr>
<td></td>
<td>When you select a table name, data and meta-data appear in the DB Details view.</td>
</tr>
<tr>
<td></td>
<td>Note: 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:

| SESSIONS | Displays the information for the current database session. |

**Related Reference**

- MS SQL Server DB Details view
OpenEdge database

You can view OpenEdge database schema by using the following views:

- OpenEdge DB Details view
- OpenEdge DB 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 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 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>• <strong>Columns</strong> - A list of all columns and related SQL information.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Indexes</strong> - A list of indexes in the selected table.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Primary key</strong> - 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>• <strong>Foreign key</strong> - A list of any foreign key restraints defined for a selected SQL table.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Preview</strong> - A subset of the records in the selected table. You can set the number of rows returned in the <strong>Preferences</strong> page.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Row count</strong> - The total number of rows in the selected table.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> 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.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Privileges</strong> - A list of both table and column privileges (as a toggle).</td>
</tr>
<tr>
<td></td>
<td>• <strong>OpenEdge Table</strong> - A list of table attributes. Attributes are extracted from the _File system table.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The Area and Can-* fields are available only for users who have DBA privileges.</td>
</tr>
<tr>
<td></td>
<td>• <strong>OpenEdge Columns</strong> - 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 <strong>Name</strong> label.</td>
</tr>
<tr>
<td></td>
<td>• <strong>OpenEdge Indexes</strong> - A list of index attributes.</td>
</tr>
</tbody>
</table>
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.
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 Progress DB Navigator to display or hide schemas. By default, catalogs are displayed while schemas are hidden from view. See Progress DB Navigator Preferences 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 Progress DB Navigator preferences, all catalogs appear below the database node. If you have selected Show schema in the Progress 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>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYSTEM TABLE</td>
<td>Displays virtual systems tables and other metaschema tables. These tables cannot be altered.</td>
</tr>
<tr>
<td>TABLE</td>
<td>Displays all application tables. Table nodes expand to show columns, indexes, and triggers.</td>
</tr>
<tr>
<td></td>
<td>Note: 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. 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.</td>
</tr>
<tr>
<td>SEQUENCES</td>
<td>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.</td>
</tr>
<tr>
<td>TRIGGERS</td>
<td>Displays ABL schema triggers, replication triggers, and Java triggers.</td>
</tr>
</tbody>
</table>
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 commit schema change upon completion of wizard.

The three Edit Schema wizards are:

- Table wizard
- Columns wizard
- Index 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>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Table Name</strong></td>
<td>A table name that is unique within the database. Table names must follow these conventions:</td>
</tr>
<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>
<tr>
<td><strong>Area</strong></td>
<td>A storage area from a list of storage areas defined for the database.</td>
</tr>
<tr>
<td><strong>Dump File</strong></td>
<td>A unique filename for dumping table contents or definitions.</td>
</tr>
<tr>
<td><strong>Label</strong></td>
<td>The label that is used in error messages. This field is followed by a <strong>Field String Attribute</strong> field for controlling text preferences and space allocation.</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>A description of the contents or purpose of the table for documentation purposes.</td>
</tr>
<tr>
<td><strong>Validation</strong></td>
<td>An expression that 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))</td>
</tr>
<tr>
<td><strong>Message</strong></td>
<td>The error message that is displayed when a record deletion is invalid. This field is followed by a <strong>Field String Attribute</strong> field for controlling text preferences and space allocation.</td>
</tr>
<tr>
<td><strong>Hidden</strong></td>
<td>When selected, creates hidden tables.</td>
</tr>
<tr>
<td><strong>Keep wizard open after adding table</strong></td>
<td>Keep the wizard open to add one or more other tables.</td>
</tr>
</tbody>
</table>
Related Tasks

Viewing and maintaining OpenEdge schema
Managing OpenEdge database tables
Connecting to a database

Related Reference

Columns wizard
Index wizard
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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>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 &amp;.</td>
</tr>
<tr>
<td><strong>Data Type</strong></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. <strong>Caution</strong>: Changing the data type from INTEGER to INT64 is irreversible.</td>
</tr>
<tr>
<td><strong>Size</strong></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><strong>Scale</strong></td>
<td>(Available only if <strong>Data Type</strong> 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>
<tr>
<td><strong>Format</strong></td>
<td>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 <strong>Field String Attribute</strong> field for controlling text preferences and space allocation. Each data type has a default format defined in the <strong>Progress DB Navigator preferences</strong>.</td>
</tr>
<tr>
<td><strong>Label</strong></td>
<td>The column heading that appears on screens or in printed reports to identify the column. The Unknown value (?) causes the string that appears in <strong>Name</strong> to be used. This field is followed by a <strong>Field String Attribute</strong> field for controlling text preferences and space allocation. <strong>Note</strong>: If you use ? as a label, precede it with a backslash. For example: ?.</td>
</tr>
<tr>
<td><strong>Col Label</strong></td>
<td>A heading that appears when a program lists data in columns. ? causes the string that appears in <strong>Label</strong> 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><strong>Init Value</strong></td>
<td>The initial value, based on data type, for the column. This field is followed by a Field String Attribute 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.  
You can find OpenEdge manuals in the Product Documentation section of the Progress Software Developer's Network website. |
| **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. |
| **Mandatory** | When selected, makes the column mandatory. That is, it cannot have the Unknown value (?). |
| **Max Size** | *(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. |
| **Code page** | *(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.
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:

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Copy</strong></td>
<td>Allows one or more columns to be created from other columns in any table. Upon selecting a column and then selecting <strong>OK</strong>, the column information is copied into the <strong>Column</strong> 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.</td>
</tr>
<tr>
<td><strong>Remove</strong></td>
<td>Removes selected columns from the tree view.</td>
</tr>
<tr>
<td><strong>Add</strong></td>
<td>Adds a new column node to the tree view and resets the fields in the <strong>Columns</strong> wizard to their initial values.</td>
</tr>
</tbody>
</table>

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**.

**Related Tasks**

- Viewing and maintaining OpenEdge database schema
- Working with OpenEdge columns
- Connecting to a database

**Related Reference**

- Table wizard
- Index wizard
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>Area</td>
<td>A storage area from a list of storage areas defined for the database.</td>
</tr>
<tr>
<td>Description</td>
<td>A description of the index for documentation purposes.</td>
</tr>
<tr>
<td>Primary</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>Active</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>Unique</td>
<td>When selected, indicates that every index key must be unique (for example, a social security number).</td>
</tr>
<tr>
<td>Word Index</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. (Note that a word index cannot be a primary index.)</td>
</tr>
<tr>
<td>Fields</td>
<td>(Add Index only) A list of table fields and index fields that allows you add or remove fields from the index.</td>
</tr>
<tr>
<td>Descending</td>
<td>(Add Index only) When selected, rows are sorted in decreasing order.</td>
</tr>
<tr>
<td>Ascending</td>
<td>(Add Index only) When selected, rows are sorted in increasing order. (This option is selected by default.)</td>
</tr>
</tbody>
</table>

**Related Concept**

SQL terminology

**Related Tasks**

Viewing and maintaining OpenEdge database schema
Indexes
Connecting to a database
Related Reference

Table wizard
Columns wizard
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>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of a new or existing sequence. Sequence names must follow certain conventions.</td>
</tr>
<tr>
<td>Initial Value</td>
<td>The first integer in the sequence. The default initial value is zero.</td>
</tr>
<tr>
<td>Increment by</td>
<td>A positive or negative integer value that changes the sequence value at each iteration. When you enter a negative value, the <strong>Upper Limit</strong> label changes to <strong>Lower Limit</strong>.</td>
</tr>
<tr>
<td>Upper/Lower Limit</td>
<td>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 <strong>Increment by</strong> 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>A value that defines the limit of the sequence. When the sequence reaches this value, it starts again at the value defined in <strong>Initial Value</strong>.</td>
</tr>
</tbody>
</table>

**Related Task**

*Working with OpenEdge database sequences*

**Related Reference**

For more information about sequences, see the ABL manuals in the **OpenEdge Product Documentation** section of the Progress Communities Web site.
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 ASSGN (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, REPPLICATION-CREATE, REPPLICATION-DELETE, REPPLICATION-WRITE, and ASSGN (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.
Related Task

- Working with OpenEdge database triggers

Related Reference

For more information about triggers, see the ABL manuals in the OpenEdge Product Documentation section of the Progress Communities Web site.
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.

Related Reference

Table wizard
Columns wizard
Index wizard
Oracle database

You can view Oracle database schema by using the following views:

- Oracle DB Details view
- Oracle DB 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>- <strong>Columns</strong> - A list of all columns and related SQL information.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Indexes</strong> - A list of indexes in the selected table.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Primary key</strong> - A list of the primary keys defined for the selected table.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Foreign key</strong> - A list of any foreign key restraints defined for a selected SQL table.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Preview</strong> - A subset of the records in the selected table. You can set the number of rows returned in the <strong>Preferences page</strong>.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Row count</strong> - The total number of rows in the selected table.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Privileges</strong> - A list of both table and column privileges (as a toggle).</td>
</tr>
<tr>
<td></td>
<td>- <strong>Dependent Objects</strong> - A list of objects dependent on that table, identified by owner, type, and name.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Status</strong> - Status details, including properties and their values.</td>
</tr>
</tbody>
</table>
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:

| **SECTIONS** | 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 Progress DB Navigator to display or hide schemas. By default, catalogs are displayed while schemas are hidden from view. See Progress DB Navigator Preferences 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 Progress DB Navigator preferences, all catalogs appear below the database node. If you have selected Show schema in the Progress DB Navigator preferences, all schemas appear below the database node.

Beneath the schema and catalog nodes, the following child nodes appear:

<p>| <strong>SYNONYM</strong> | 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. |
| <strong>TABLE</strong> | 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 sort after the uppercase table names. |
| <strong>VIEW</strong> | 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. |
| <strong>SEQUENCE</strong> | Displays all sequences, which create an incremented, serial list of numbers applied to a column in a database table. |</p>
<table>
<thead>
<tr>
<th>PACKAGE</th>
<th>Displays all packages, which contain specific procedures, functions, and other schema items in one entity.</th>
</tr>
</thead>
<tbody>
<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:

| SESSIONS | Displays information about the current session, such as username, Session ID, machine name, and the user's logon time, in the **DB Details view**. |
| JOBS | Displays information about a job, or task, that you run once or on a particular schedule, in the **DB Details view**. |
| TOP SQL | Displays SQL executions in the **DB Details view**, enabling you to see how resources are affected when a SQL statement is executing. |

**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>

**Related Reference**

- Oracle DB Details view
- Progress DB Navigator preferences
Preferences

You can access the preference pages for Progress DB Navigator by choosing Windows > Preferences > Progress DB Navigator.

The preference pages are:

- Progress DB Navigator
- Connection Profiles
- Databases
  - OpenEdge Schema Editing
- JDBC Drivers
- SQL Editor
Progress DB Navigator preferences

The following list describes the preferences you can set in the **Progress DB Navigator Preferences** page:

<table>
<thead>
<tr>
<th>Preference</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Preview row limit</strong></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><strong>SQL Results row limit</strong></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><strong>SQL Results rows to batch</strong></td>
<td>Specifies the number of rows to batch when retrieving records. The default is 100.</td>
</tr>
<tr>
<td><strong>SQL Editor separator</strong></td>
<td>Specifies the separator to use for separating multiple SQL statements. The default is a semicolon.</td>
</tr>
<tr>
<td><strong>Auto commit</strong></td>
<td>Determines the default <strong>AutoCommit</strong> mode for a connection. If you select the <strong>Auto commit</strong> option in the Progress DB Navigator preferences, <strong>AutoCommit</strong> is selected by default for each SQL connection profile. (You can, however, select or deselect <strong>AutoCommit</strong> for any connection profile individually.) 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 option</strong> selected for the connection profile.</td>
</tr>
<tr>
<td><strong>Commit On Close</strong></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><strong>Columns auto-completing assistance</strong></td>
<td>Enables auto-completion for columns within the <strong>SQL Editor</strong>. 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><strong>Save and restore last selected node in Database Structure View</strong></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><strong>Append table name to column</strong></td>
<td>Causes the table name to be appended when column names are pasted to the clipboard (for example, <code>Customer.Name</code>).</td>
</tr>
</tbody>
</table>
| **Enclose in quotes** | Causes quotes to be added to column names, table names, or schema when copied to the clipboard (for example, "CustNum", "Name") when generating SQL select statements or creating a table script.

Note: Column names that contain leading underscores (for example, _Buffer-ID) must be enclosed by quotes. Otherwise, the SQL Editor does not recognize them. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Delimiter</strong></td>
<td>Specifies the character used as a delimiter when multiple objects are copied to the clipboard.</td>
</tr>
</tbody>
</table>
| **Show catalog or Show schema** | If Show catalog 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 Show schema is selected, all of the schemas are listed beneath the root node.
When you change from one preference to the other, you must refresh the DB Structure view, or close and reopen the connection. |

**Related Task**
Setting preferences

**Related Reference**
OpenEdge Schema Editing preferences
SQL Editor preferences
SQL connection profile preferences
SQL Connection profile preferences

The **Connection Profiles** preferences page displays SQL connection profiles. This page includes the following controls:

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Adds a new SQL connection profile for the workspace.</td>
</tr>
<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>

**Related Task**
Setting preferences

**Related Reference**
OpenEdge Schema Editing preferences
SQL Editor preferences
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:

| **Add**          | 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]`. See the documentation for the driver if you require more information about the driver. |
| **Edit**        | Modify aspects of an existing JDBC driver. Select the driver you want to edit from the list of available drivers, then click **Edit**. |
| **Copy**        | Copy a driver. Select the driver you want to copy from the list of available drivers, then click **Copy**. |
| **Remove**      | Delete a JDBC driver. Select the driver you want to delete from the list of available drivers, then click **Remove**. |
| **Set Default** | Set the selected JDBC driver as the default. |

**Related Tasks**

- Setting preferences
- Configuring a JDBC driver
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><strong>Open script in editor upon completion of wizard</strong></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>
<tr>
<td><strong>Execute and commit Schema change on completion of wizard</strong></td>
<td>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. <strong>Note:</strong> This option can be disabled only if the Open script in editor option is selected.</td>
</tr>
<tr>
<td><strong>Grant public SQL rights</strong></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><strong>Default Character Size</strong></td>
<td>Establishes the default size for new character fields.</td>
</tr>
<tr>
<td><strong>Default Decimal Size</strong></td>
<td>Establishes the default size for new decimal fields.</td>
</tr>
<tr>
<td><strong>Update format to reflect entered size value</strong></td>
<td>When this option is enabled, you can enter a value in the Size to Format ratio 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><strong>Default Format</strong></td>
<td>Displays default formats for all ABL data types. You can change the defaults by selecting a data type and clicking the Set button.</td>
</tr>
</tbody>
</table>

**Related Task**

Setting preferences

**Related Reference**

Edit Schema wizards
SQL Editor Preferences
SQL Connection profile preferences
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.

Related Task
SQL scripting

Related Reference
SQL Editor
SQL Connection profile preferences
OpenEdge AppBuilder

OpenEdge AppBuilder enables the visual design of user interfaces in an OpenEdge project by launching the OpenEdge AppBuilder embedded in an Eclipse view.

When you run AppBuilder embedded in Eclipse, you use the OpenEdge Architect Editor to edit .w files, instead of the AppBuilder section editor.

Topics in this section include:

- Starting AppBuilder
- Starting AppBuilder with Dynamics enabled
- Working with .w files
- Enabling AppBuilder shortcut keys
Starting AppBuilder

OpenEdge AppBuilder is a multipurpose application development environment that supports a broad, integrated range of application and development options. However, you will probably run AppBuilder in OpenEdge Architect primarily as a visual editor for the GUI portion of your project.

After opening the AppBuilder perspective (Window>Open Perspective>OpenEdge AppBuilder), use one of the following methods to start an AppBuilder session:

- Click the AppBuilder button on the toolbar.
- Select OpenEdge>Tools>AppBuilder from the menu bar.
- Double-click on a .w file in the Resources view.

When the AppBuilder starts, the AppBuilder main window, the Palette, and PRO*Tools appear in the AppBuilder Designer view.

Usage Guidelines

There are some differences between running the AppBuilder as a standalone application and running it embedded in OpenEdge Architect:

- The AppBuilder section editor is not supported Open Edge Architect. Use the ABL Editor to make changes to the .w file.
- The ABL Editor displays the .w file, while the AppBuilder Designer displays a visual rendering of the .w file. Changes are reflected both in the ABL Editor and in the AppBuilder Designer. See Working with .w files.
- Within the OpenEdge Architect, some AppBuilder functionality (compile, code preview, and options) is disabled.
- Some AppBuilder options (for example, File>Save and Debug) use OpenEdge Architect.
- When opening a file through the AppBuilder, keep in mind that the file must belong to the corresponding project in order to have an editor associated with it.
- AppBuilder help can be invoked from the Help menu of AppBuilder. It is not included in the OpenEdge Architect Guide in Eclipse help.
- 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 OpenEdge Architect toolbar, OpenEdge Architect runs the last saved version of whatever file has focus in the code editor. To run a file from the OpenEdge Architect toolbar that was changed in the design window, you have to save the file in the code editor.

Related Tasks

Starting AppBuilder with Dynamics enabled
Working with .w files
Starting AppBuilder with Dynamics enabled

You can use AppBuilder, running embedded in OpenEdge Architect, 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 OpenEdge Architect are:

- **Select the Progress Dynamics option during installation of OpenEdge Architect** — 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 OpenEdge Architect:

1. Select a project name in the Resources viewer of OpenEdge Architect.
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.

   There is a default configuration file for GUI application development in OpenEdge-install-dir/gui/dynamics.

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.
**Related Reference**

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.
Working with .w files

Creating new .w files

To create .w files, use the New button from the AppBuilder perspective. Before you can edit new files with the OpenEdge Architect ABL Code Editor, you must add them to a project.

Opening .w files

Open .w files by using one of the following methods:

- Drag and drop the .w file from the Resources view onto the AppBuilder window.
- Double-click on a .w file in the Resources view.
- Choose File>Open from the AppBuilder menubar.

If the file being opened is part of the project, an instance of the ABL Code Editor opens to allow editing of the file.

If the file is not a .w file, AppBuilder passes the Open File operation to the ABL Editor.

Editing .w files

A .w file opened in the AppBuilder is displayed graphically in the Design view. In addition, the code within the .w file is displayed below the Design view in the ABL Code 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
AppBuilder and Other OpenEdge Tools

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 OpenEdge Architect. You can also use the **Save** and **Save As** options directly from within OpenEdge Architect.

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 OpenEdge Architect 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 OpenEdge Architect toolbar, OpenEdge Architect runs the last saved version of whatever file has focus in the code editor. To run a file from the OpenEdge Architect 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.

**Related Tasks**

StartAppBuilder
Enabling AppBuilder shortcut keys

Most of the shortcut keys in OpenEdge Architect differ from AppBuilder shortcut keys. If you want to use AppBuilder shortcut keys, you can apply many of the AppBuilder key bindings by setting a single workspace preference.

To enable AppBuilder shortcut keys:

1. Open the Preferences dialog 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 example:

![Preferences dialog](image)

Notice 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 OpenEdge Architect.

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**.
If you are running OpenEdge Architect in an external Eclipse workbench (an Eclipse workbench other than the one that ships with OpenEdge Architect), some configuration is required before you can enable AppBuilder shortcut keys:

1. In Windows Explorer, navigate to `OpenEdge-install-dir/oeide/eclipse/plugins` (the plug-ins folder of the default OpenEdge Architect workbench).
2. 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.
3. Open `../eclipse/configuration/config.ini` in the additional Eclipse installation with a text editor.

When you restart the external Eclipse workbench, you can enable AppBuilder shortcut keys.

Related Reference

See the Keys topic in the Workbench User Guide.
Other OpenEdge Tools

From OpenEdge Architect, you can also access ProTools, the Procedure Editor, the Data Dictionary, and many other OpenEdge tools that are not Eclipse applications.

Most of these tools do not run within OpenEdge Architect, however. When started, they run in a window outside the Eclipse framework.

The exceptions are WebSpeed WebTools and OpenEdge Explorer, which run in embedded browsers.

Online help for these tools is available from a tool's user interface and not from OpenEdge Architect's help menu.

Topics in this section include:

- Starting the Audit Policy Maintenance Tool
- Starting the Data Administration Tool
- Starting the Data Dictionary
- Starting the OpenEdge Explorer
- Starting the Procedure Editor
- Starting ProTools
- Starting WebSpeed WebTools
Starting the Audit Policy Maintenance Tool

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 OpenEdge Architect perspective except the Progress DB Navigator and the OpenEdge Debugger.

When started, the Audit Policy Maintenance tool runs in a window outside of the OpenEdge Architect Eclipse framework. Online help is available from the tool's menu bar.
Starting the Database Administration Tool

The Database Administration Tool is a 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 Architect perspective except Progress DB Navigator and OpenEdge Debugger.

When started, the Database Administration tool runs in a window outside the OpenEdge Architect Eclipse framework. Online help is available from the tool's menu bar.
Starting the Data Dictionary

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 OpenEdge Architect 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 Architect perspective except Progress DB Navigator and OpenEdge Debugger.

When started, the Data Dictionary runs in a window outside the OpenEdge Architect Eclipse framework. Online help is available from the tool's menu bar.
Starting OpenEdge Explorer

OpenEdge Explorer, like Progress 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 OpenEdge Architect 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.
Starting the Procedure Editor

The Procedure Editor is an OpenEdge ABL code editor.

**Note:** When you run the Procedure Editor from OpenEdge Architect, 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 **Resources** view, the selected file opens in the Procedure Editor.

When started, the Procedure Editor runs in a window outside the OpenEdge Architect Eclipse framework. Online help is available from the tool's menu bar.
**Starting PRO*Tools**

PRO*Tools is a set of utility programs that help you develop and run OpenEdge applications. They include AppServer Session Information, Color Changer, Configuration Information, Database Connections, Font Changer, OS Prompt, Procedure Object Viewer, COM Object Viewer, ProtoGen, Run Procedure, PROPATH Editor, Screen Capture, Screen Scaling, Session Attributes, SmartObjects Conversion Utility, Static SmartObject, Dynamic Object Conversion Utility, Window Viewer, and Windows Information.

Start PRO*Tools by selecting **OpenEdge>Tools>PRO*Tools** from the menu bar in any OpenEdge Architect perspective except the Progress DB Navigator and the OpenEdge Debugger. The PRO*Tools toolbar appears containing icons to launch the utility programs.

When started, the PRO*Tools utilities run in windows outside of OpenEdge Architect Eclipse framework. Online help is available from a utility’s menu bar.
Starting WebSpeed WebTools

You can use the OpenEdge Architect **WebSpeed** view to run WebSpeed WebTools or other WebSpeed applications. Start the view by selecting **Window>Show view>WebSpeed**.

The **WebSpeed** view is actually an embedded Web browser that has an Address bar where you can start WebTools by entering the WorkShop URL. Use this format:

http://host_name/scripts_dir/messenger/WService=broker/workshop

To run WebTools applications you need access to a Web server that is running WebSpeed in development mode. OpenEdge Architect does have an internal Web server, but WebTools functionality is limited when running on the internal Web server.

Online help is available from a help icon on each WebTools page.

**Related Reference**

**WebSpeed Settings**

For more information about WebSpeed, see the WebSpeed manuals in the Product Documentation section of the Progress Software Developer's Network Web site.
Introducing the Meta Catalog

The OpenEdge Architect’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 Architect 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**

- ABL annotations
- Access to catalog data
- Content builders
- Local and master catalogs

**Related Reference**

- ABL Annotation Syntax
- Meta Catalog database
Concepts

This section includes the following topics:

ABL annotations
Access to catalog data
Content builders
Local and master catalogs
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.

**Related Tasks**
- Adding annotations to source files
- Adding Sonic ESB annotations

**Related Reference**
- ABL Annotation content builder
- ABL annotation syntax
- Annotation Generator
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.

**Related Concepts**
- Introducing the Meta Catalog

**Related Tasks**
- Accessing the Meta Catalog Explorer
- Searching the Meta Catalog

**Related Reference**
- Meta Catalog Explorer view
- Meta Catalog Search view
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.

OpenEdge Architect 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 OpenEdge Architect**

OpenEdge Architect provides the following predefined content builders:

<table>
<thead>
<tr>
<th>ABL Annotation</th>
<th>Extracts annotations that you either manually add to the source code or included through the Annotation Generator.</th>
</tr>
</thead>
<tbody>
<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>

**Related Concepts**

Introducing the Meta Catalog

**Related Tasks**

Adding custom file extensions to content builders

**Related Reference**

ABL Annotation content builder
ABL annotation syntax
ABL Application Schema content builder
ABL Reference content builder
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:

| Local          | 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. |
| Master         | 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. |

Related Concepts
Introducing the Meta Catalog

Related Tasks
Building a master catalog
Linking to a master catalog

Related Reference
OECATALOG utility
Tasks

This section includes help on the following tasks:

Accessing the Meta Catalog Explorer
Adding custom file extensions to content builders
Adding new files to a catalog
Building a master catalog
Changing catalog definitions
Creating a local catalog
Creating templates for the Annotation Generator
Linking to a master catalog
Rebuilding a local catalog
Searching the Meta Catalog
Setting Meta Catalog preferences
Using the Meta Catalog Explorer
Accessing the Meta Catalog Explorer

To access the Meta Catalog Explorer view:

1. Choose **Window>Show View>Other...** to display the **Show View** dialog box.
2. Choose **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.

**Related Concepts**

- Access to catalog data

**Related Tasks**

- Using the Meta Catalog Explorer

**Related Reference**

- Meta Catalog Explorer view
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 OpenEdge Architect>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.

Related Concepts

Content builders

Related Tasks

Changing catalog definitions
Setting Meta Catalog preferences

Related Reference

Meta catalog preferences
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 Resources 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.

Related Tasks
  - Changing catalog definitions
  - Rebuilding a local catalog
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 the Progress Explorer. Remember that the database server has to support SQL.
   e. Start the database server.

2. Run the New Catalog wizard
   a. Start an OpenEdge Architect 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 browse 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.

**Related Concepts**

Local and master catalogs

**Related Tasks**

Creating a local catalog
Linking to a master catalog

**Related Reference**

OECATALOG utility
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 **OpenEdge Architect>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.

**Related Tasks**
- Adding custom file extensions to content builders
- Rebuilding a local catalog

**Related Reference**
- Meta Catalog Preferences
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>**.

   **Note:** 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.

   **Note:** You can change the definition of a catalog later through the **Preferences** window.

**Related Concepts**
Local and master catalogs

**Related Tasks**
- Adding new files to a catalog
- Building a master catalog
- Changing catalog definitions
- Rebuilding a local catalog

**Related Reference**
Meta Catalog preferences
Creating templates for the Annotation Generator

1. Choose **Window>Preferences**.
2. Choose **OpenEdge Architect>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.

   **Note:** The dialog does not check the syntax of the template.

6. Choose **OK** to save the template.

**Related Concepts**

ABL annotations

**Related Tasks**

Adding annotations to source files
Adding Sonic ESB annotations

**Related Reference**

ABL Annotation content builder
ABL annotation syntax
Annotation Generator
Linking to a master catalog

1. Choose File>New>Other to display the New dialog box.
2. Choose OpenEdge>Meta Catalog>Meta Catalog 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.

   **Note:** 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.

**Related Concepts**
Local and master catalogs

**Related Tasks**
Adding new files to a catalog
Building a master catalog
Changing catalog definitions
Rebuilding a local catalog

**Related Reference**
Meta Catalog preferences
OECAATALOG utility
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** &gt; **Preferences** to open the **Preferences** window.
2. Choose **OpenEdge Architect** &gt; **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.

**Related Concepts**
- Local and master catalogs
- Content builders

**Related Tasks**
- Adding new files to a catalog
- Creating a local catalog

**Related Reference**
- Meta Catalog preferences
Searching the Meta Catalog

1. Choose **Search>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.

   **Note:** 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.

   **Note:** If the same file's contents have been loaded into two catalogs, the search results show duplicate entries for that data.

**Related Concepts**

- Access to catalog data

**Related Tasks**

- Accessing the Meta Catalog Explorer
- Using the Meta Catalog Explorer

**Related Reference**

- Meta Catalog Search view
Setting Meta Catalog preferences

1. Choose **Window>Preferences** to open the **Preferences** window.
2. Choose **OpenEdge Architect>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.

**Related Concepts**

- Introducing the Meta Catalog

**Related Tasks**

- Changing catalog definitions
- Adding custom file extensions to content builders

**Related Reference**

- Meta Catalog preferences
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.

**Related Concepts**

Access to catalog data

**Related Tasks**

Accessing the Meta Catalog Explorer
Searching the Meta Catalog

**Related Reference**

ABL Annotation content builder
ABL Application Schema content builder
ABL Reference content builder
Meta Catalog Explorer view
Reference

This section includes help on the following reference topics:

- ABL Annotation content builder
- ABL Annotation syntax
- ABL Application Schema content builder
- ABL Reference content builder
- Annotation Generator
- Meta Catalog database
- Meta Catalog Explorer view
- Meta Catalog preferences
- Meta Catalog Search view
- OECATALOG utility
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:
**Related Concepts**

- Content builders
- ABL Annotations

**Related Tasks**

- Adding custom file extensions to content builders
- Creating templates for the Annotation Generator
- Adding annotations to source files
- Adding Sonic ESB annotations

**Related Reference**

- ABL Annotation Syntax
- Annotation Generator
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:

```plaintext
@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.

**Related Concepts**
- Content builders
- ABL annotations

**Related Tasks**
- Creating templates for the Annotation Generator
- Adding annotations to source files
- Adding Sonic ESB annotations

**Related Reference**
- ABL Annotation content builder
- Annotation Generator
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
Meta Catalog

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:

```sql
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></td>
</tr>
<tr>
<td></td>
<td></td>
<td>LABEL</td>
<td>&quot;Name&quot;</td>
<td></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:
Related Concepts
Content builders

Related Tasks
Adding custom file extensions to content builders
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:

<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>
</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:

```plaintext
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:
**Related Concepts**

Content builders

**Related Tasks**

Adding custom file extensions to content builders
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**

**Add Annotations dialog**

### Annotation Generation preference page

You can access the preference page by choosing **Window>Preferences** and then selecting the **OpenEdge Architect>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. |

**Return to top**
Add Annotations dialog

You can access dialog by choosing File>New>Other and then selecting the OpenEdge>Editor>Annotation Generation node.

<table>
<thead>
<tr>
<th>Select annotation template combo-box</th>
<th>Select an existing annotation template. The annotation text displays in the editor and can be customized.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annotation text editor</td>
<td>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. <strong>NOTE:</strong> The editor does not check the syntax of the annotation.</td>
</tr>
<tr>
<td>Available Resource filter</td>
<td>Enables you to filter the Available Resource treeview. Choose the Filter button to apply a filter.</td>
</tr>
<tr>
<td>Available Resource treeview</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>Backup all selected resources toggle</td>
<td>When selected causes files to backup before applying the annotation. By default, this option is turned on.</td>
</tr>
<tr>
<td>Location</td>
<td>The directory where the backup files are stored. <strong>NOTE:</strong> 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>Finish</td>
<td>Starts the Annotation Generator using the selected options.</td>
</tr>
</tbody>
</table>

Return to top

Related Concepts

ABL Annotations

Related Tasks

Adding annotations to source files
Adding Sonic ESB annotations

Related Reference

ABL Annotation content builder
ABL annotation syntax
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.
- Consider creating a secondary login broker to handle the SQL connections. For more information on this, see Progress Solution #P7843, "How to configure a secondary SQL-92 Broker for Progress 9.1D and above", on the Knowledge Center.

For detailed information on database startup parameters, see OpenEdge Data Administration: Database Administration on the OpenEdge Documentation website.

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.

**Related Concepts**
- Introducing the Meta Catalog
- Local and master catalogs
Related Tasks

- Changing catalog definitions
- Rebuilding a local catalog
- Adding new files to a catalog
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 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>

Related Concepts
Access to catalog data

Related Tasks
Accessing the Meta Catalog Explorer
Using the Meta Catalog Explorer

Related Reference
ABL Annotation content builder
ABL Application Schema content builder
ABL Reference content builder
Meta Catalog Search view
Meta Catalog preferences

You can access the Meta Catalog Preferences by choosing **Window>Preferences** and then selecting the **OpenEdge Architect>Meta Catalog** node.

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>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>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.</strong></td>
<td></td>
</tr>
<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><strong>New</strong></td>
<td>Launches the <strong>Create Meta Catalog</strong> 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><strong>Add Master</strong></td>
<td>Launches the <strong>Add Meta Catalog</strong> 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><strong>Remove</strong></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><strong>Import</strong></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>
<tr>
<td><strong>Export All</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Rebuild</strong></td>
<td>Rebuilds the selected catalog. This option is disabled for master catalogs.</td>
</tr>
<tr>
<td><strong>Note:</strong> 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.</td>
<td></td>
</tr>
</tbody>
</table>

[Back to top.](#)
# 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></td>
<td>▪ <strong>On save of element</strong> - 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>
</tr>
<tr>
<td></td>
<td></td>
<td>▪ <strong>On rebuild</strong> - 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>
</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 Architect’s internal version of the catalog.xml file.</td>
<td></td>
</tr>
<tr>
<td>Database information tab</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>Meta Catalog database path</td>
<td>The full path of the physical catalog database, including the filename.</td>
</tr>
<tr>
<td></td>
<td>Service/Port for server (-S)</td>
<td>The connection port or service name for the catalog's database server.</td>
</tr>
<tr>
<td></td>
<td>Additional server parameters</td>
<td>Any necessary startup parameters for the database.</td>
</tr>
<tr>
<td></td>
<td>Automatically start/stop server toggle box</td>
<td>Whether the catalog's server should automatically start or stop when you enter or leave this workspace.</td>
</tr>
<tr>
<td><strong>Connection information tab</strong></td>
<td><strong>Host</strong></td>
<td>Name of the host where the catalog is.</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td><strong>Service/Port</strong></td>
<td></td>
<td>The connection port or service for the catalog's database server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td></td>
<td>The logical name of the database.</td>
</tr>
<tr>
<td><strong>User</strong></td>
<td></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><strong>Password</strong></td>
<td></td>
<td>The password to use connecting to the database.</td>
</tr>
</tbody>
</table>

| **Mapping of root path tab** | **Root path** | 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. |
|                             | **Local path** | The path to your local copies of the resources in a master catalog. |
|                             | **Edit** | 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. |

| **Content Builders tab** | **Code** | 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. |
|                         | **Description** | The description of the content builder. |

| **Projects tab** | **Project name** | 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. |
**Content Builders page**

<table>
<thead>
<tr>
<th><strong>Code</strong></th>
<th>The content builder's code.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>The description of the content builder.</td>
</tr>
<tr>
<td><strong>Class</strong></td>
<td>The Java class for the content builder.</td>
</tr>
<tr>
<td><strong>Extensions</strong></td>
<td>The file extensions against which the content builder should run.</td>
</tr>
<tr>
<td><strong>New...</strong></td>
<td>Launches a dialog for entering a new content builder definition.</td>
</tr>
<tr>
<td><strong>Edit...</strong></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><strong>Remove</strong></td>
<td>Removes a content definition from this workspace. This does not delete the content builder class file.</td>
</tr>
<tr>
<td><strong>Restore Defaults</strong></td>
<td>Deletes the current definitions and replaces them with the default definitions of the predefined content builders that ship with the Architect.</td>
</tr>
<tr>
<td><strong>Apply</strong></td>
<td>Saves any edits for the selected content builder to the Architect's internal version of the catalog.xml file.</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.

Back to top.

**Related Concepts**
- Introducing the Meta Catalog
- Local and master catalogs

**Related Tasks**
- Changing catalog definitions
- Creating a local catalog
- Adding custom file extensions to content builders
- Setting Meta Catalog preferences
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><strong>Projects to search</strong></th>
<th>Lists all projects in your workspace. Select the projects whose contents you want to search.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Catalogs to search</strong></td>
<td>Lists all catalogs defined for your workspace. Select the catalogs whose contents you want to search.</td>
</tr>
<tr>
<td><strong>Search criteria</strong></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><strong>Content Type</strong></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><strong>Entry Type</strong></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><strong>Entry</strong></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><strong>Attribute</strong></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><strong>Attribute Value</strong></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><strong>AND/OR</strong></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><strong>Add</strong></td>
<td>Adds a line to the bottom of the criteria in the browse.</td>
</tr>
<tr>
<td><strong>Insert</strong></td>
<td>Inserts a line above the currently selected line in the browse.</td>
</tr>
<tr>
<td><strong>Remove</strong></td>
<td>Removes the selected line in the browse.</td>
</tr>
</tbody>
</table>
**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.

**Save**
Saves the current search criteria under the name in the **Saved Criteria** combo-box.

**Remove**
Removes a saved criterion from the file.

**Search**
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.

---

**Related Concepts**
Access to catalog data

**Related Tasks**
Searching the Meta Catalog

**Related Reference**
Meta Catalog Explorer view
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>` is the catalog definitions file, usually called `catalog.xml`,

`<catalog-name>` is the name of a specific definition in the definitions file,

`<root-path>` is the root path of a project in the catalog,

`<new-path>` is the new root path where the project has been moved,

`<resource-path>` is the path, either absolute or relative to the `-root`, where the resource that you want to scan exists,

`<project>` is 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.

**Notes**

- 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 the OpenEdge Architect, 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 ( " ).

**Related Concepts**

- Local and master catalogs

**Related Tasks**

- Building a master catalog
Introducing Tools for Business Logic

OpenEdge Architect Tools for Business Logic 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.

Related Concepts

Application modeling
Design principles and the OpenEdge Reference Architecture
The Business Logic tool set
Concepts

This section includes the following topics:

- Application modeling
- Design principles and the OpenEdge Reference Architecture
- The Business Logic tool set
- Component types
- Design output
- Component diagrams
- Component models
- Model storage options
Application modeling

In the context of OpenEdge Architect, 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.

Model storage

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.

Enforcement of component standards

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.

OpenEdge Architect 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.

Related Concepts

Design output
Model storage options

**Related Tasks**

- Modeling new components
- Generating code from a model
- Creating diagrams from code
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.

Related Concepts

- Application modeling
- Component types
The Business Logic tool set

Two OpenEdge Architect 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 OpenEdge 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 OpenEdge 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.)

**Related Concepts**
- Component Designer
- Application Schema Explorer

**Related Tasks**
- Modeling components

**Related Reference**
- Component Designer
- Application Schema Explorer
- Outline view for diagrams
Component types

You can use Tools for Business Logic to create and edit the following component types:

- Temp tables
- 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

**Related Concepts**

- Application modeling
- The Business Logic tool set

**Related Tasks**

- Modeling components

**Related Reference**

- Component Designer
ProDataSets

A ProDataSet is a pre-defined view of data 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

Related Concepts

Temp tables
Application modeling
The Business Logic tool set

Related Tasks

Modeling components

Related Reference

Component Designer
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.

This section includes the following topics:

- Component diagrams
- Component models
- Model storage options
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.

Related Concepts

Component models

Related Tasks

Modeling components
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.

Related Concepts

- Model storage options
- Component diagrams

Related Tasks

- Modeling components
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 OpenEdge Architect 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.

**Related Concepts**

Application modeling

**Related Tasks**

Setting Tools for Business Logic preferences
Tasks

This section includes the following topics:

- Setting Tools for Business Logic preferences
- Model database management
- Modeling components
- Matching diagrams to ABL code
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 OpenEdge Architect and then Tools for Business Logic.

This section includes the following:

Choosing a background color
Controlling automatic linking
Determining how models are stored
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 **OpenEdge Architect** 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**.

**Related Concepts**

*The Business Logic tool set*

**Related Tasks**

*Setting Tools for Business Logic preferences*

**Related Reference**

*Tools for Business Logic options*
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 OpenEdge Architect and select Tools for Business Logic.
3. Click the Auto link components check box to toggle the option off or on.

Related Concepts

The Business Logic tool set

Related Tasks

Adding relationships
Setting Tools for Business Logic preferences

Related Reference

Tools for Business Logic options
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 **OpenEdge Architect** 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.

**Related Concepts**
- Model storage options

**Related Tasks**
- Creating a model database
- Editing database settings

**Related Reference**
- Tools for Business Logic options
Model database management

This section explains how to create a model database, edit its settings, and share connection information. It includes the following:

Creating a model database
Sharing remote database settings
Editing database settings
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,
   - or
   - Select Window> Preferences. In the left pane of the Preferences window, expand OpenEdge Architect 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.

   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 OpenEdge Architect 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.

   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.

10. 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 and/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 nec-
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.

11. 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:** OpenEdge Architect does not support the use of multiple component model databases. You can create only one per workspace.

**Related Concepts**
- Model storage options

**Related Tasks**
- Editing database settings
- Setting Tools for Business Logic preferences

**Related Reference**
- Create Component Model Database wizard
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.

To export database information:

1. Define the remote connection parameters, and optionally the project associations, as explained in Creating a model database.
2. Select Window>Preferences>OpenEdge Architect>Tools for Business Logic>Component Model Database.
3. Click Export.
4. 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.

Other developers can then import the exported settings.

To import exported database settings:

1. Select Window>Preferences>OpenEdge Architect>Tools for Business Logic>Component Model Database.
2. Click Import.
3. At the Import Component Database file browser, navigate to the location of the exported .xml file. Select the file and click Open.

Related Concepts
Model storage options

Related Tasks
Creating a model database
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:

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.

Related Concepts
Model storage options

Related Tasks
Creating a model database

Related Reference
Database settings
Modeling components

This section explains the process of creating Business Logic components. It includes the following topics:

Creating a new component
Editing a component
Creating a new component

Complete the following steps to create a new Business Logic component:

1. Select File>New>Other>OpenEdge>Tools for Business Logic>Business Logic Component and click Next.
2. Enter a folder in your workspace as the location for the component diagram.
3. In the Component name field, enter a file name with a .dgm extension.
4. Select one of the valid component types.
5. Click Finish or Next, whichever button is enabled.

   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.

6. 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.

   Use one or more of the following methods to populate the model list:

   - Click Create. Then specify the name and location of a new .t4bl file, and click OK.
   - 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.
   - Click Browse File System 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.

   To remove a model from the list, select it and click Remove.

7. 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.

**Related Concepts**

- Component types
- Application modeling
- The Business Logic tool set
- Design output

**Related Tasks**

- Editing a component
Creating diagrams from code
Determining how models are stored

Related Reference
Outline view for diagrams
Editing a component

The topics in this section explain how to define the content of a component:

- Adding new data structures
- Using existing data structures
- Identifying application schema elements
- Adding relationships
- Adding diagram notes
- Adding annotations
- Deleting components
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.

Related Concepts

The Business Logic tool set

Related Tasks

Using existing data structures
Adding relationships
Adding diagram notes
Adding annotations
Deleting components
Related Reference

Component Designer
Using existing data structures

As you build new components, OpenEdge Architect 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
- Source code

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.

To use the Outline view:

1. In the Component Designer, open the diagram to which you want to add existing elements.
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.

   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.

To use the palette:

1. Click an item in the **Existing Components** drawer of the palette.
2. 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.
3. Select the element that you want to use, and click **OK**. The view window closes.
4. 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 can-
Tools for Business Logic

Copying 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.

For more information about the DB Structure view, see the Database Navigator volume in OpenEdge Architect help.

Copying from source code

You can 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. To accomplish this:
   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.

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.
**Related Concepts**

The Business Logic tool set

**Related Tasks**

Identifying application schema elements

**Related Reference**

Component Designer
Outline view for diagrams
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.

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.

Related Tasks

Using existing data structures

Related Reference

Application Schema Explorer
Outline view for diagrams
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.

**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.

### 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.

**Related Concepts**
- Component types

**Related Tasks**
- Controlling automatic linking

**Related Reference**
- Component Designer
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.

Related Reference

Component Designer
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 OpenEdge Architect help for detailed information about catalogs and annotations.

To add an annotation to a component:

1. Create or edit a diagram.
2. In the Other drawer of the Component Designer palette, click New Annotation.
3. 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:

```plaintext
@annotation_name [(attribute = "value",attribute = "value")...]
```

To edit the values:

1. Select the annotation in the diagram.
2. Open the Properties view and click in the Value column to the right of Annotation Properties (under Attributes). An ellipsis button ( ) appears at the right side of the column. Click this button to open the Annotation Properties dialog.
3. Enter the annotation name in the field at the top.
4. Click the text in the Key column and enter the attribute name.
5. Click the text in the Value column and enter the attribute value.
6. Use the New entry and Remove entry buttons to add more attributes or to delete previously entered attributes.
7. Click OK.

**Related Tasks**

Generating code from a model
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.
   
   To delete the components from both the model and the diagram, press Shift + Delete.

Note: You cannot delete the base component of the diagram. You can delete only those elements that you add after creating the base component.

Related Concepts

Component diagrams
Component models

Related Tasks

Editing a component

Related Reference

Component Designer
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. The topics in this section explain these procedures:

Validating a component diagram
Generating code from a diagram
Creating diagrams from 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.

**Related Tasks**

Generating code from a diagram

**Related Reference**

Component Designer
Generating code from a diagram

You can produce source code that defines the model represented in a component diagram.

By default, OpenEdge Architect 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:

1. Edit the component diagram in the Component Designer.
2. Right-click on the canvas and select Generate Code from the menu.
3. OpenEdge Architect 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.

Customizing 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.

Related Tasks

Validating a component diagram
Creating diagrams from code

Related Reference

Component Designer
Creating diagrams from code

OpenEdge Architect 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.
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:
   - Check or clear the box in the **ObjectName** column to specify whether you want a diagram created for the component.
   - Edit the name of the diagram file.

   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

**Related Concepts**

Component diagrams

**Related Tasks**

Generating code from a diagram

**Related Reference**

Code to Model Component wizard
Reference

This section contains the following topics:

Component Designer
Application Schema Explorer
Outline view for diagrams
Tools for Business Logic options
Create Component Model Database wizard
New Business Logic Component wizard
Code to Model Component wizard
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 ( ) at the top of the Palette bar. The symbol becomes a right arrow ( ), 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 ( ) 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.

**Related Concepts**

The Business Logic tool set

**Related Tasks**

Modeling new components
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 OpenEdge Architect 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.

**Related Concepts**

- Model storage options
- Application modeling
- Component types

**Related Tasks**

- Identifying application schema elements
- Generating code from a model
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 it 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 (¶). 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.

**Related Concepts**

- The Business Logic tool set

**Related Tasks**

- Using existing data structures
- Modeling components

**Related Reference**

- Component Designer
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>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>The folder in which diagram (.dgm) files are to be created.</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>
</tr>
<tr>
<td>Component Type</td>
<td>The type of Business Logic component.</td>
</tr>
<tr>
<td>Component Name</td>
<td>The name of the diagram file to be created (editable).</td>
</tr>
<tr>
<td>Commands</td>
<td>Browse To open a standard file browser that you can use to specify a location for the diagram files.</td>
</tr>
<tr>
<td></td>
<td>Select All To select all components.</td>
</tr>
<tr>
<td></td>
<td>Deselect All To deselect all components.</td>
</tr>
<tr>
<td></td>
<td>Finish To create diagrams for the selected components.</td>
</tr>
<tr>
<td></td>
<td>Cancel To close the wizard without creating any new components.</td>
</tr>
</tbody>
</table>

**Related Concepts**
- Component diagrams
- Component types
- Application modeling

**Related Tasks**
- Creating diagrams from code
- Creating a new component
Tools for Business Logic options

This section explains the pages of the Context-sensitive help Preferences dialog. To access these settings, select **Window>Preferences>OpenEdge>Tools for Business Logic**.

This section includes the following:

- Component Designer options
- Database list
- Database settings
Component Designer options

These options govern basic behavior of the Component Designer. You open this dialog by selecting **Window>Preferences>OpenEdge Architect>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>Not currently used; do not modify.</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>

**Related Concepts**
- Design output

**Related Tasks**
- Setting Tools for Business Logic preferences
- Adding relationships

**Related reference**
- Component Designer
Database list

This dialog identifies your component model database, if any. You open the dialog by selecting **Window>Preferences>OpenEdge Architect>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>Auto-start</td>
<td></td>
<td>The setting (true or false) of the option to start the database server automatically when you launch OpenEdge Architect 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>

**Related Concepts**

- Model storage options

**Related Tasks**

- Model database management
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>OpenEdge Architect>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</td>
<td>The logical name for the database.</td>
</tr>
<tr>
<td>(read-only field)</td>
<td></td>
</tr>
<tr>
<td>Component model database path</td>
<td>The complete path name for the database <code>.db</code> 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 OpenEdge Architect 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 <strong>localhost</strong> 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:

<table>
<thead>
<tr>
<th>Commands</th>
<th>Restore Defaults</th>
<th>To discard changes to database settings and revert to default values.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apply</td>
<td></td>
<td>To save the current database settings and keep the Preferences dialog open.</td>
</tr>
<tr>
<td>OK</td>
<td></td>
<td>To save all current values and close the Preferences dialog.</td>
</tr>
<tr>
<td>Cancel</td>
<td></td>
<td>To close the Preferences dialog without saving any changes.</td>
</tr>
</tbody>
</table>

Related Concepts

- Component models
- Model storage options

Related Tasks

- Creating a model database
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.

This section includes the following:

- Database name
- Local database path and server
- Database connection parameters
- Project association
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>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>

**Related Concepts**
- Application models
- Model storage options

**Related Tasks**
- Creating a model database

**Related Reference**
- Component Designer options
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>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Component model database path</th>
<th>The complete path name for the database .db file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service/port for server (-S)</td>
<td>The name or port to use for the database server.</td>
</tr>
<tr>
<td>Additional server parameters</td>
<td>Any valid 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 OpenEdge Architect and stop when you quit.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Commands</th>
<th>Back</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>To return to the preceding page of the Create Component Model Database wizard.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Commands</th>
<th>Next</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>To proceed to specify database connection parameters.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Commands</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>To create the local database with default connection parameters and without associating projects.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Commands</th>
<th>Cancel</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>To close the Create Component Model Database wizard without creating a database.</td>
</tr>
</tbody>
</table>

**Related Concepts**
- Application models
- Model storage options

**Related Tasks**
- Creating a model database

**Related Reference**
- Component Designer options
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:

<table>
<thead>
<tr>
<th>Fields</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host</strong></td>
<td>The name of the remote database host machine, or <strong>localhost</strong> for a local database.</td>
</tr>
<tr>
<td><strong>Service/Port</strong></td>
<td>The name or port for the database server.</td>
</tr>
<tr>
<td><strong>Database</strong></td>
<td>The logical name of the database.</td>
</tr>
<tr>
<td><strong>User</strong></td>
<td>A valid database user name. For a local database, your login name is used by default.</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>The password for the specified user, if required.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Commands</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Back</strong></td>
<td>To return to the preceding page of the Create Component Model Database wizard.</td>
</tr>
<tr>
<td><strong>Next</strong></td>
<td>To proceed to associate projects with the model database.</td>
</tr>
<tr>
<td><strong>Cancel</strong></td>
<td>To close the Create Component Model Database wizard without saving any changes.</td>
</tr>
</tbody>
</table>

**Related Concepts**
- Application models
- Model storage options

**Related Tasks**
- Creating a model database

**Related Reference**
- Component Designer options
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:

<table>
<thead>
<tr>
<th>Fields</th>
<th>Available Projects</th>
<th>Selected Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The list of your existing projects in the current workspace.</td>
<td>The list of projects that you have selected to associate with the model database.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Commands</th>
<th>Add</th>
<th>Remove</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>To move the projects selected in the Available Projects field to the Selected Projects field.</td>
<td>To move the projects selected in the Selected Projects field to the Available Projects field.</td>
</tr>
</tbody>
</table>

| Remove All     | To move all projects in the Selected Projects field to the Available Projects field. |
| Add All        | To move all projects in the Available Projects field to the Selected Projects field. |
| Back           | To return to the preceding page of the Create Component Model Database wizard.     |
| Finish         | To complete the creation of the model database.                                    |
| Cancel         | To close the Create Component Model Database wizard without creating a database.  |

**Related Concepts**
- Application models
- Model storage options

**Related Tasks**
- Creating a model database
**Related Reference**

- Component Designer options
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**.

This section includes the following:

- **New component name and location**
- **Model file selection**
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>Component name</td>
<td>The file name of the diagram (.dgm) file</td>
</tr>
<tr>
<td>Select a</td>
<td>Radio buttons for specifying the type of Business Logic component to create (Temp table or ProDataSet).</td>
</tr>
<tr>
<td>Component Type</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Commands</th>
<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>
<td></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, <code>appModel.t4bl</code>, or with your component model database.</td>
<td></td>
</tr>
<tr>
<td>Cancel</td>
<td>To close the wizard without creating a new component.</td>
<td></td>
</tr>
</tbody>
</table>

**Related Concepts**
- Component types
- Design output

**Related Tasks**
- Creating a new component
- Determining how models are stored
- Creating a model database

**Related Reference**
- Component Designer options
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>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Commands</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Create</td>
<td>To display the Create File dialog, where you can specify a name and location for a new model file.</td>
</tr>
<tr>
<td>Browse Workspace</td>
<td>To display a Resource Selection dialog, where you can specify a folder in your workspace to search for all .t4bl files.</td>
</tr>
<tr>
<td>Browse File System</td>
<td>To display a standard file browser with a *.t4bl filter, enabling you to browse for and select a model anywhere in your file system.</td>
</tr>
<tr>
<td>Remove</td>
<td>To remove the .t4bl file selected in the T4BL model files field from the list (but not from the file system).</td>
</tr>
<tr>
<td>Back</td>
<td>To return to the preceding page of the New Business Logic Component wizard.</td>
</tr>
<tr>
<td>Finish</td>
<td>To create the specified component diagram and associate it with the specified model file.</td>
</tr>
<tr>
<td>Cancel</td>
<td>To close the wizard without creating a new component.</td>
</tr>
</tbody>
</table>

Related Concepts

Component types
Design output
Related Tasks

- Creating a new component
- Component models
- Determining how models are stored

Related Reference

- Component Designer options
Introducing OpenEdge Architect customization

You can customize the following aspects of your OpenEdge Architect 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 OpenEdge Architect, to define menu options and the output generated when those options are used.

- **Startup routines** - You can define ABL procedures that run automatically when OpenEdge Architect starts.

- **Event subscription** - OpenEdge Architect 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.

**Note**: Customizations for menus, toolbars, and templates are stored in a separate Eclipse plugin, by default com.openedge.pdt.extensibility.user. Although there should be no need to specify a different plugin for this purpose, you can do so if necessary on the Customization preferences page.

**Related Concepts**
- Menu and toolbar customization
- Template customization
- Startup customization
- Event subscription

**Related Tasks**
- Using the Customization Editor
- Customizing project startup
- Subscribing to OpenEdge Architect events
Concepts

This section includes the following topics:

Menu and toolbar customization
Template customization
Startup customization
Event subscription
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.

Related Tasks

Using the Customization Editor

Related Reference

Customization Editor
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.

OpenEdge Architect 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, OpenEdge Architect 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 must have access to valid JET (Java Emitter Template) files to take advantage of these customization features. The Customization Editor creates the appropriate wizards and menu entries, but OpenEdge Architect does not provide template-creation tools.

**Note:** Any JET templates that you use 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:

- **Progress Software Developers Network Product Documentation section:** See *OpenEdge Architect: Working with Custom Templates*
  - Introduction to JET (Eclipse Corner Article)
  - *EMFT JET Developer Guide*

Related Tasks

- Customizing file templates

Related Reference

- Customization Editor
Startup customization

You can write an ABL procedure to run automatically when you start OpenEdge Architect. For example, you might create a startup procedure that subscribes to OpenEdge Architect events.

Related Tasks

Customizing OpenEdge Architect startup
Event subscription

OpenEdge Architect 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.

Related Tasks

Subscribing to OpenEdge Architect events
Tasks

This section includes the following topics:

- Setting customization preferences
- Using the Customization Editor
- Customizing project startup
- Subscribing to OpenEdge Architect events
Setting customization preferences

To open the **Customization** preferences page, select **Window>Preferences>OpenEdge Architect>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, and template 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 Id** 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**.

**Related Concepts**

- Menu and toolbar customization
- Template customization

**Related Reference**

- Customization Preferences dialog
Using the Customization Editor

This section includes the following topics:

- Adding menu and toolbar commands
- Customizing code templates
- Editing and deleting options
- Sharing customization settings
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, Extensibilit&y, pressing Alt+Y opens the menu.)

3. Click Add.

   **Note:** As a shortcut to bypass 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 (File>Save or ).

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 at the confirmation prompt that appears. (You may need to click Yes twice.) If you skip this step, your changes do not take effect until you restart OpenEdge Architect.

**Related Concepts**
- Launch configurations

**Related Tasks**
- Editing and deleting options

**Related Reference**
- Customization Editor
Customizing code templates

If you have access to custom Java Emitter Templates (JET templates) for creating ABL code, you can use the Customization Editor to add file-creation wizards that use these templates to the File>New menu. 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.

OpenEdge Architect does not provide template-creation tools. 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.

Note: 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.

Adding a file-creation wizard

Follow these steps to add a new entry to the File>New menu:

1. In the Available Extension Points tree view, click New Templates.
2. Click Add.
3. Enter a name, and optionally an image file for an accompanying icon, to appear on the File>New menu.
4. In the Template file field, enter the name of the Java class file containing the compiled JET template.
5. Save your changes.

Adding a template override

Follow these steps to specify a template for use in place of a standard OpenEdge Architect template:

1. In the Available Extension Points tree view, click Template Overrides.
2. Click Add.
3. In the Template field, select the standard template that you want to override from the drop-down list.
4. In the Template file field, enter the name of the Java class file containing the compiled JET template to be used instead.
5. Save your changes.

Saving changes

After adding or overriding template definitions:
1. Save your changes with **File>Save** or ![Save button].

2. To make the changes effective immediately, click ![Reset perspective button] (rightmost on the Customization Editor toolbar) to reset the OpenEdge Editor perspective. Then click **Yes** at the confirmation prompt that appears. (You may need to click **Yes** twice.)

**Related Concepts**

- Template customization

**Related Tasks**

- Editing and deleting options
- Setting customization preferences

**Related Reference**

- Customization Editor
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 (**File > Save** or click ).

5. To make the changes effective immediately, click (rightmost on the Customization Editor toolbar) to reset the OpenEdge Editor perspective. Then click **Yes** at the confirmation prompt that appears. (You may need to click **Yes** twice.) If you skip this step, your changes do not take effect until you restart OpenEdge Architect.

**Related Tasks**

- Adding menu and toolbar commands
- Customizing code templates

**Related Reference**

- Customization Editor
Sharing customization settings

To share custom menu, toolbar, and template settings with another user, provide to that user:

- 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\arcitect\eclipse\plugins\com.openedge.pdt.extensibility.user_version_number

- Copies of all custom templates.

The other user should then do the following:

- 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.

- Save the custom templates in the desired location, and make sure that location is in the Java class path.

**Related Concepts**

- Menu and toolbar customization
- Template customization

**Related Reference**

- Customization Preferences page
Customizing project startup

When you open an OpenEdge Architect 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 OpenEdge Architect events.
Subscribing to OpenEdge Architect events

Each time any of various OpenEdge Architect 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 OpenEdge Architect events.

Output parameters for oeide_event

The following output parameters are published:

<table>
<thead>
<tr>
<th>eventName</th>
<th>One of the following values:</th>
</tr>
</thead>
<tbody>
<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>
</tbody>
</table>

<table>
<thead>
<tr>
<th>projectName</th>
<th>The name of the project in which the event occurs.</th>
</tr>
</thead>
<tbody>
<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 OpenEdge Architect Debugger does not trigger publication of an event.

AppBuilder events

When AppBuilder runs in OpenEdge Architect, most of its events are published by both OpenEdge Studio and OpenEdge Architect. Whenever possible, it is recommended that you use the OpenEdge Architect events, rather than the corresponding OpenEdge Studio events. However, the following OpenEdge studio events do not have OpenEdge Architect counterparts:

- New
- Before-Open
Customization

Before-Close
Before-Check-Syntax
Check-Syntax
Before-Check-Syntax-Partial
Check-Syntax-Partial

**Named event publishing for AppServers in OpenEdge Architect**

OpenEdge Architect 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</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>
</tbody>
</table>
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>
</tbody>
</table>
Sample event-subscription startup routine

The following example shows a startup file, _idestartup.p, that subscribes to OpenEdge Architect events and runs event_alert.p as a persistent procedure:

```c
/*-------------------------------------------------------------
File : _idestartup.p
Purpose : Subscribe to OE Architect events, call event_alert.p
-------------------------------------------------------------
*/
DEFINE VARIABLE mySubscribeHandle AS HANDLE NO-UNDO.
RUN event_alert.p PERSISTENT SET mySubscribeHandle.
SUBSCRIBE PROCEDURE mySubscribeHandle TO "oeide_event" ANYWHERE.
```

Each time an event occurs, event_alert.p displays an alert with the values of the three
oeide_event parameters:

/*-----------------------------------------------
----------
    File     : event_alert.p
    Purpose   : Display alerts for OE Architect events
---------------------------------------------------------*/
PROCEDURE oeide_event.
DEFINE INPUT PARAMETER eventName AS CHARACTER.
DEFINE INPUT PARAMETER projectName AS CHARACTER.
DEFINE INPUT PARAMETER programName AS CHARACTER.
DEFINE INPUT PARAMETER eventData AS CHARACTER.
MESSAGE "Event name: "     eventName SKIP
    "Project name: "       projectName SKIP
    "Program name: "       programName SKIP
VIEW-AS ALERT-BOX.
END PROCEDURE.

Related Concepts

Startup customization
Reference

This section includes the following topics:

- Customization Editor
- Customization Preferences page
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 OpenEdge Architect) with custom JET templates, overriding their association with standard installed templates.

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. The tables below describe these controls.
### Controls available in all contexts

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Available Extension Points</strong></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 <strong>Menu/Toolbar Entries</strong> node.</td>
</tr>
<tr>
<td>![Add Program to Editor Menu]</td>
<td>Displays the form for a new entry under the <strong>Editor Context Menu</strong> node.</td>
</tr>
<tr>
<td>![Add Custom Template]</td>
<td>Displays the form for a new entry under the <strong>New Templates</strong> node.</td>
</tr>
<tr>
<td>![Add Template Override]</td>
<td>Displays the form for a new entry under the <strong>Template Overrides</strong> node.</td>
</tr>
<tr>
<td>![Refresh Customization Options]</td>
<td>Resets the OpenEdge Editor perspective after editing, making changes available immediately without the need to restart OpenEdge Architect. You are first prompted to save changes if necessary.</td>
</tr>
<tr>
<td><strong>Add</strong></td>
<td>Displays the form for a new entry of the type selected in the tree view.</td>
</tr>
<tr>
<td><strong>Remove</strong></td>
<td>Deletes the currently selected entry.</td>
</tr>
<tr>
<td><strong>Version</strong></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 <code>com.openedge.pdt.extensibility.user</code>) is in use. If you change the version number here, a copy of <code>com.openedge.pdt.extensibility.user</code> is created with this version number. <strong>Caution:</strong> 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>
## Controls available for Menu/Toolbar Entries or Editor Context Menu

| **Menu label** | Description
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>(Available only when top-level Menu/Toolbar Entries node is selected)</em></td>
<td>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>
</tbody>
</table>

| **Name** | Description
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></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** | Description
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>(Optional)</em></td>
<td>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</td>
</tr>
</tbody>
</table>

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** | Description
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>(Optional)</em></td>
<td>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.</td>
</tr>
</tbody>
</table>

| **Tooltip** | Description
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>(Optional)</em></td>
<td>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. Tootips have no effect in menus.</td>
</tr>
</tbody>
</table>

| **Program name** | Description
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>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:</td>
</tr>
</tbody>
</table>

- 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 available for New Templates

<table>
<thead>
<tr>
<th>Name</th>
<th>The name of the wizard. This text will appear on the <strong>File&gt;New</strong> menu.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Icon</td>
<td><em>(Optional)</em> An image to appear on the menu. The image file must be in the <strong>icons</strong> 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>

### Controls available for Template Overrides

<table>
<thead>
<tr>
<th>Template</th>
<th>The name of the standard OpenEdge Architect template be overwritten.</th>
</tr>
</thead>
<tbody>
<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>

**Related Concepts**

- Menu and toolbar customization
- Template customization
- Launch configurations

**Related Tasks**

- Using the Customization Editor
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>OpenEdge Architect>Advanced>Customization**.

The following controls are available:

<table>
<thead>
<tr>
<th>Customization plugin Id</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<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 OpenEdge Architect starts.</td>
</tr>
<tr>
<td>Restore Defaults</td>
<td>Sets the value of the <strong>Customization plugin Id</strong> to com.openedge.pdt.extensibility.user and clears the <strong>Template search locations</strong> 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>

**Related Concepts**
- Template customization

**Related Tasks**
- Customizing code templates