Progress®



# OpenEdge® Service Pack 11.7.1:
## New Information

OpenEdge®

# Copyright

**© 2017 Progress Software Corporation and/or one of 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.

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

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

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



**October 2017**

**Last updated with new content:** Release 11.7.2

**Updated: 2021/06/28**

# Table of Contents

**Preface....7**
Purpose....7
Audience....8
Organization....8
Use ABL documentation....8
References to ABL compiler and run-time features....9
References to ABL data types....9
Typographical conventions....9
Examples of syntax descriptions....10
Long syntax descriptions split across lines....12
Complex syntax descriptions with both required and optional elements....12
Example procedures....13
OpenEdge messages....13
Obtain more information about OpenEdge messages....14

**What's New in 11.7.1....15**
Installation and Configuration....15
ABL for Business Entities....16
Foreign key support....16
Transactional Submit support (behavior change)....17
OpenEdge Management....18
OpenEdge SQL....18
Progress Developer Studio for OpenEdge....19
Support for Foreign Key Annotation....19
Compile files with errors....20
Export a Web UI project....20
View the log files....21
Support for CAST function....21
Support for ANT tasks....21
RDBMS....22
Server Technologies and Tools....24

# Preface

For details, see the following topics:


- Purpose
- Audience
- Organization
- Use ABL documentation
- Typographical conventions
- Examples of syntax descriptions
- Example procedures
- OpenEdge messages


## Purpose

*OpenEdge® Service Pack 11.7.1: New Information* documents important information about the OpenEdge Release 11.7.1 Service Pack, with reference to new and existing documentation for Release 11.7 as required. Release 11.7 documentation set references might include:

- Product manuals
- HTML-based online help
- Web papers

References to these documents attempt to identify sections or topics that you can use in searches of the product documentation set.

# Audience

This guide is primarily intended for OpenEdge application developers and system administrators who have installed 11.7.1 as the most recent Service Pack for Release 11.7.

# Organization

This book contains the following:

- Installation and Configuration on page 15

  Provides information about changes to the install process that allow you to install both a service pack and a major release simultaneously, and updates to the supported Infragistics controls.

- ABL for Business Entities on page 16

  Provides information about field-level Foreign Key support, and a behavior change in Submit operation methods for Business Entity class instances generated in Progress Developer Studio for OpenEdge.

- OpenEdge Management on page 18

  Provides information about new options to manage and monitor your CPU resources in both local and remote AdminServer environments.

- OpenEdge SQL on page 18

  Provides information about support for Autonomous Schema Update on a Replication target database, and Connection Failover and Load Balancing support for Progress DataDirect Series ODBC and JDBC drivers.

- Progress Developer Studio for OpenEdge on page 19

  Provides information about new features including: Foreign Key annotation, support for the CAST function, support for ANT tasks, how to compile only files with errors, how to export a Web UI project, and how to view the log files.

- RDBMS on page 22

  Provides information about support for diagnostic data collection, and a new option for IDXCHECK.

- Server Technologies and Tools on page 24

  Provides information using an environment variable to avoid a SonicMQ broker error in Windows 10.

# Use ABL documentation

OpenEdge provides a special purpose programming language for building business applications. In the documentation, the formal name for this language is *ABL (Advanced Business Language)*. With few exceptions, all keywords of the language appear in all UPPERCASE, using a font that is appropriate to the context. All other alphabetic language content appears in mixed case.

For the latest documentation updates go to https://docs.progress.com.

## References to ABL compiler and run-time features

ABL is both a compiled and an interpreted language that executes in a run-time engine. The documentation refers to this run-time engine as the *ABL Virtual Machine (AVM)*. When the documentation refers to ABL source code compilation, it specifies *ABL* or *the compiler* as the actor that manages compile-time features of the language. When the documentation refers to run-time behavior in an executing ABL program, it specifies *the AVM* as the actor that manages the specified run-time behavior in the program.

For example, these sentences refer to the ABL compiler's allowance for parameter passing and the AVM's possible response to that parameter passing at run time: "ABL allows you to pass a dynamic temp-table handle as a static temp-table parameter of a method. However, if at run time the passed dynamic temp-table schema does not match the schema of the static temp-table parameter, the AVM raises an error." The following sentence refers to run-time actions that the AVM can perform using a particular ABL feature: "The ABL socket object handle allows the AVM to connect with other ABL and non-ABL sessions using TCP/IP sockets."

## References to ABL data types

ABL provides built-in data types, built-in class data types, and user-defined class data types. References to built-in data types follow these rules:

- Like most other keywords, references to specific built-in data types appear in all `UPPERCASE`, using a font that is appropriate to the context. No uppercase reference ever includes or implies any data type other than itself.
- Wherever *integer* appears, this is a reference to the `INTEGER` or `INT64` data type.
- Wherever *character* appears, this is a reference to the `CHARACTER`, `LONGCHAR`, or `CLOB` data type.
- Wherever *decimal* appears, this is a reference to the `DECIMAL` data type.
- Wherever *numeric* appears, this is a reference to the `INTEGER`, `INT64`, or `DECIMAL` data type.

References to built-in class data types appear in mixed case with initial caps, for example, `Progress.Lang.Object`. References to user-defined class data types appear in mixed case, as specified for a given application example.

# Typographical conventions

This documentation uses the following typographical and syntax conventions:

| Convention | Description |
|---|---|
| **Bold** | Bold typeface indicates commands or characters the user types, provides emphasis, or the names of user interface elements. |
| *Italic* | Italic typeface indicates the title of a document, or signifies new terms. |
| **SMALL, BOLD CAPITAL LETTERS** | Small, bold capital letters indicate OpenEdge key functions and generic keyboard keys; for example, **GET** and **CTRL**. |

| Convention | Description |
|---|---|
| **KEY1+KEY2** | A plus sign between key names indicates a **simultaneous** key sequence: you press and hold down the first key while pressing the second key. For example, **CTRL+X**. |
| **KEY1 KEY2** | A space between key names indicates a **sequential** key sequence: you press and release the first key, then press another key. For example, **ESCAPE H**. |
| **Syntax:** | |
| `Fixed width` | A fixed-width font is used in syntax, code examples, system output, and file names. |
| *`Fixed-width italics`* | Fixed-width italics indicate variables in syntax. |
| ***`Fixed-width bold`*** | Fixed-width bold italic indicates variables in syntax with special emphasis. |
| `UPPERCASE fixed width` | ABL keywords in syntax and code examples are almost always shown in upper case. Although shown in uppercase, you can type ABL keywords in either uppercase or lowercase in a procedure or class. |
| Period (.) or colon (:) | All statements except `DO`, `FOR`, `FUNCTION`, `PROCEDURE`, and `REPEAT` end with a period. `DO`, `FOR`, `FUNCTION`, `PROCEDURE`, and `REPEAT` statements can end with either a period or a colon. |
| **[ ]** | Large brackets indicate the items within them are optional. |
| [ ] | Small brackets are part of ABL. |
| **{ }** | Large braces indicate the items within them are required. They are used to simplify complex syntax diagrams. |
| { } | Small braces are part of ABL. For example, a called external procedure must use braces when referencing arguments passed by a calling procedure. |
| **\|** | A vertical bar indicates a choice. |
| **...** | Ellipses indicate repetition: you can choose one or more of the preceding items. |

# Examples of syntax descriptions

In this example, `ACCUM` is a keyword, and *`aggregate`* and *`expression`* are variables:

### Syntax

```
ACCUM aggregate expression
```

FOR is one of the statements that can end with either a period or a colon, as in this example:

```
FOR EACH Customer NO-LOCK:
  DISPLAY Customer.Name.
END.
```

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

### Syntax

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

In this example, the outer (small) brackets are part of the language, and the inner (large) brackets denote an optional item:

### Syntax

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

A called external procedure must use braces when referencing compile-time arguments passed by a calling procedure, as shown in this example:

### Syntax

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

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

### Syntax

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

In this example, you must include two expressions, and optionally you can include more. Multiple expressions are separated by commas:

### Syntax

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

In this example, you must specify MESSAGE and at least one *expression* or SKIP [ ( *n* ) ], and any number of additional *expression* or SKIP [ ( *n* ) ] is allowed:

### Syntax

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

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

### Syntax

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

## Long syntax descriptions split across lines

Some syntax descriptions are too long to fit on one line. When syntax descriptions are split across multiple lines, groups of optional and groups of required items are kept together in the required order.

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

### Syntax

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

## Complex syntax descriptions with both required and optional elements

Some syntax descriptions are too complex to distinguish required and optional elements by bracketing only the optional elements. For such syntax, the descriptions include both braces (for required elements) and brackets (for optional elements).

In this example, ASSIGN requires either one or more *field* entries or one *record*. Options available with *field* or *record* are grouped with braces and brackets:

### Syntax

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

# Example procedures

OpenEdge documentation may provide example code that illustrates syntax and concepts. You can access many of the example files, and details for installing them, from the following locations:

- A self-extracting Documentation and Samples file available on the OpenEdge download page of the Progress Software Download Center
- The OpenEdge Product Documentation Overview page on Progress Communities:

https://community.progress.com/technicalusers/w/openedgegeneral/1329.openedge-product-documentation-overview.aspx

Once installed, you can locate the example files in the following paths under the OpenEdge Documentation and Samples installation directory:

| This directory . . . | Contains examples for the following documents . . . |
|---|---|
| `src\prodoc\dotnetobjects` | *OpenEdge Development: GUI for .NET Programming* |
| `src\prodoc\dynamics` | The Progress Dynamics documentation |
| `src\prodoc\getstartoop` | *OpenEdge Development: Object-oriented Programming* |
| `src\prodoc\handbook` | *OpenEdge Getting Started: ABL Essentials* |
| `src\prodoc\interfaces` | *OpenEdge Development: Programming Interfaces* |
| `src\prodoc\json` | *OpenEdge Development: Working with JSON* |
| `src\prodoc\langref` | *OpenEdge Development: ABL Reference* |
| `src\prodoc\prodatasets` | *OpenEdge Development: ProDataSets* |
| `src\prodoc\visualdesigner` | *OpenEdge Getting Started: Introducing Progress Developer Studio for OpenEdge Visual Designer* |
| `src\prodoc\xml` | *OpenEdge Development: Working with XML* |
| `src\samples\open4gl\java` | *OpenEdge Development: Java Open Client* |

# OpenEdge messages

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

- **Execution messages** inform you of errors encountered while OpenEdge is running a procedure; for example, if OpenEdge cannot find a record with a specified index field value.
- **Compile messages** inform you of errors found while OpenEdge is reading and analyzing a procedure before running it; for example, if a procedure references a table name that is not defined in the database.

- **Startup messages** inform you of unusual conditions detected while OpenEdge is getting ready to execute; for example, if you entered an invalid startup parameter.

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

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

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

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

If you encounter an error that terminates OpenEdge, note the message number before restarting.

# Obtain more information about OpenEdge messages

In Windows platforms, use OpenEdge online help to obtain more information about OpenEdge messages. Many OpenEdge tools include the following Help menu options to provide information about messages:

- Choose **Help** > **Recent Messages** to display detailed descriptions of the most recent OpenEdge message and all other messages returned in the current session.
- Choose **Help** > **Messages** and then type the message number to display a description of a specific OpenEdge message.
- In the Procedure Editor, press the **HELP** key or **F1**.

On UNIX platforms, use the OpenEdge `pro` command to start a single-user mode character OpenEdge client session and view a brief description of a message by providing its number.

**To use the pro command to obtain a message description by message number**:

1. Start the Procedure Editor:

   ```
   OpenEdge-install-dir/bin/pro
   ```

2. Press **F3** to access the menu bar, then choose **Help** > **Messages**.
3. Type the message number and press **ENTER**. Details about that message number appear.
4. Press **F4** to close the message, press **F3** to access the Procedure Editor menu, and choose **File** > **Exit**.

# 1

# What's New in 11.7.1

This topic describes the new features included in OpenEdge Release 11.7.1.

For details, see the following topics:

- Installation and Configuration
- ABL for Business Entities
- OpenEdge Management
- OpenEdge SQL
- Progress Developer Studio for OpenEdge
- RDBMS
- Server Technologies and Tools

## Installation and Configuration

Installing the Release 11.7.1 service pack includes the following updates:

- **Installing a service pack** — If you are installing a major release and one of its service packs, for example, 11.7 and 11.7.1, you can install both using a single installation package.

  If you do not have OpenEdge installed, the service pack installation installs both the major release and the service pack.

  If you have a major release of OpenEdge already installed, the service pack installation detects the existing installation and you can choose to only install the changes in the service pack.

- **Infragistics** — OpenEdge release 11.7.1 uses Infragistics NetAdvantage for .NET v2016 Vol 2.CLR 4.0. The Infragistics controls are upgraded to version 16.2.20162.2084.

---

**Attention:**

**Configuration Issues for Progress Application Server for OpenEdge**

You should be aware that an installation of a service pack release does not overwrite all PAS for OpenEdge configuration files. This is done so that you will not lose customizations you may want to preserve.

However, in order to take advantage of new or changed features, you may need to manually update configuration files in the core server (*oe_install_dir*`/servers/pasoe`) and all of its instances after installing the service pack. For more information, see the `README.patch.txt` file, located in the *oe_install_dir*`/servers/pasoe/patch` directory after the service pack installation.

---

# ABL for Business Entities

Release 11.7.1 includes the following ABL enhancements for Business Entities:

- **Foreign key support** — Support is available in Business Entities to allow certain web app configurations generated in Kendo UI® Builder by Progress® to look up a unique record in a parent temp-table based on a foreign key field value in a child temp-table. You can add this support using a field-level Foreign Key annotation that defines both the foreign key on the child and how it is used to access the parent temp-table.
- **Transactional Submit support** — A behavior change in this release affects how the Submit operation works when a Business Entity class is generated using Progress Developer Studio for OpenEdge with the `OpenEdge.BusinessLogic.BusinessEntity` abstract class specified as its base class. In prior releases, the generated Submit operation method, by default, handles each record change as a separate transaction on the server (referred to as a *Bulk* Submit). With this release, the Submit operation method, by default, handles all record changes together in a single transaction on the server (referred to as a *Transactional* Submit). In addition, a manual coding option is available to restore the behavior in prior releases for a Submit operation generated for such a Business Entity.

## Foreign key support

Specify the field-level Foreign Key annotation on a child temp-table for which you want to define a look up in a given parent temp-table using the following syntax:

```
@openapi.openedge.entity.foreignkey ( [ name="foreign-key-name", ]
fields="child-field", parent="parent-temp-table", parentFields="parent-field" ).
```

*foreign-key-name*

The name of this foreign key on the associated child temp-table. If you do not specify the `name` attribute to specify a foreign key name, the value you specify for *parent-temp-table* (see below) is used to create a foreign key name.

*child-field*

A single field in the child temp-table that defines the *foreign key*. This field must have a data type that is compatible with a field in the parent temp-table whose value must match to satisfy the look up.

*parent-temp-table*

A reference to the parent temp-table, which you specify as *resource*.*parentTTname*, where *resource* is the annotated name of the Data Object resource that defines the temp-table, and *parentTTname* is the name of the parent temp-table, defined in the Business Entity that implements that resource.

*parent-field*

A single field in the parent temp-table that defines a *candidate* (or *primary*) *key*. This field must represent a value in the parent temp-table, on which a unique match can be found with the foreign key value in the child temp-table.

Foreign Key annotations have the following additional requirements and options:

- The Foreign Key annotation syntax identifies a `fields` attribute for the child temp-table and a `parentFields` attribute for the parent temp-table, implying that multiple fields can be specified to define a foreign key and its corresponding candidate key. However in Release 11.7.1, only a single field is supported to define both the foreign key on the child and the corresponding candidate key on the parent.
- The Data Object resource that defines the parent temp-table can be a different resource from the one that defines the child temp-table. However, both resources must be provided by the same Data Object Service. That is, they must be defined in the same Data Service Catalog.
- If the resource that defines the parent temp-table is for a ProDataSet, this temp-table must be a top-level temp-table in that ProDataSet. In other words, it cannot be a child temp-table in any data-relationship defined for the ProDataSet.
- The parent temp-table can be the same as the child temp-table. That is, the look up from child in the parent can be recursive.
- You can specify more than one Foreign Key annotation on a given child temp-table so look ups can be performed in multiple parent temp-tables using different foreign keys defined on the same child temp-table.

You can also specify Foreign Key annotations for a Business Entity using the **Edit Annotations** page of the **Define Service Interface** wizard in Progress Developer Studio for OpenEdge. For more information, see Support for Foreign Key Annotation on page 19.

For more information on using foreign keys to define lookups for web apps in Kendo UI Builder, see the following Kendo UI Builder by Progress documentation:

- *Kendo UI Builder by Progress: Modernizing OpenEdge Applications*
- *Kendo UI Builder by Progress: Using Kendo UI Designer*

# Transactional Submit support (behavior change)

In prior releases, the Submit operation implemented by any generated Business Entity that inherits the OpenEdge Business Entity base class, `BusinessEntity.cls` in `OpenEdge.BusinessLogic.pl`, handles each record change (create, update, or delete) for the Submit operation as a separate transaction, for as many transactions as there are record changes in this Bulk Submit operation mode. Therefore, the server potentially manages multiple transactions for the single network request that handles the Submit operation.

In Release 11.7.1, the base Business Entity class has been updated to use a Transactional Submit as the default mode for any generated Business Entity class that implements a Submit operation. This new default mode handles all record changes for the Submit operation as a single transaction on the server. Therefore, the server manages one transaction for the single network request that handles the Submit operation and all of its record changes.

If you need the Submit operation to function in the default Bulk Submit mode of prior releases, you can call the `SUPER:SaveRows( )` method instead of the `SUPER:Submit( )` call generated in the Submit operation method of your Business Entity class.

For example, you might generate a Business Entity class with the following default Submit operation method:

```
METHOD PUBLIC VOID SubmitdsCustomer(INPUT-OUTPUT DATASET dsCustomer):

    SUPER:Submit(DATASET dsCustomer BY-REFERENCE).

END METHOD.
```

You can then replace the `SUPER:Submit( )` call with the `SUPER:SaveRows( )` call as follows:

```
METHOD PUBLIC VOID SubmitdsCustomer(INPUT-OUTPUT DATASET dsCustomer):

    SUPER:SaveRows(DATASET dsCustomer BY-REFERENCE).

END METHOD.
```

# OpenEdge Management

Release 11.7.1 of OpenEdge Management includes the following:

- **New options to manage CPU resources** — OpenEdge Management monitors many CPU resources, and managing these CPU resources spread across local and remote AdminServer environments can be complex. OpenEdge Management release 11.7.1 provides two new options for better management of CPU resources available in both environments:
  - **System Wide** — Monitors only the usage of system-wide CPU resource and removes all other CPU resources available for an AdminServer.
  - **Individual CPU Monitoring** — Monitors the usage of individual CPU resources available for an AdminServer, including the usage of system-wide CPU resource.

  These options allow you to choose how you want to manage and monitor your CPU resources.

  For more information about changing resource monitoring settings, see *OpenEdge Management and OpenEdge Explorer: Getting Started.*

# OpenEdge SQL

Release 11.7.1 of OpenEdge SQL includes support for the following features:

- **Autonomous Schema Update on Replication target database**

Autonomous Schema Update(ASU) functions differently when working with a read-only Replication target database. Because the Replication target database is read-only, the schema cannot be updated. In the event of an SQL width problem caused by SQL reporting applications on the Replication target, ASU will create DDL commands on the Replication target database to address the SQL width issues. These DDL statements are ALTER statements and the syntax is as follows:

```
ALTER TABLE table-name ALTER COLUMN column-name SET PRO_SQL_WIDTH column_width;
```

These statements are written to a log file when ASU is enabled on the Replication target DB. The log file name will follow the naming convention `SQL_asu_alter_commands_serverpid_timestamp_A.log`. A second log file is named `SQL_asu_alter_commands_serverpid_timestamp_B.log`. The log files have a size limit of 500MB, and once the size limit is exceeded on the first log file, logging happens in the second log file. Once the second log file size exceeds 500MB, the first log file will be overwritten. These files are shared across sessions for the same database. Users should then manually run the DDL commands from the log file on the Replication source database using a tool like SQL Explorer. Only users with SQL DBA access on the Replication source can perform these actions.

---

**Note:**

Logging is on by default and users can choose to either turn logging on or off using the following statements:

```
SET PRO_ASU_REPLICATION_SERVER LOG ON;
```

to turn logging on and

```
SET PRO_ASU_REPLICATION_SERVER LOG OFF;
```

to turn off logging.

---

- **Connection Failover and Load Balancing support for Progress DataDirect Series ODBC and JDBC drivers**

  OpenEdge SQL supports Connection Failover and Load Balancing provided by Progress DataDirect Connect Series for ODBC and JDBC drivers. SQL client products have these drivers embedded. This enables the SQL clients to use the Connection Failover and Load Balancing features of the Progress DataDirect Connect Series for ODBC and JDBC drivers.

---

**Note:** For more information on Progress DataDirect Connect Series for ODBC and JDBC drivers, see *Progress DataDirect Documentation*.

---

# Progress Developer Studio for OpenEdge

Release 11.7.1 of Progress Developer Studio for OpenEdge includes many updates.

## Support for Foreign Key Annotation

Progress Developer Studio for OpenEdge now allows you to add Foreign key as an annotation type when you add field-level annotations on the **Edit Annotation** page. The section that follows describes the details of the page.

The **Edit Annotations** page of the **Define Service Interface** wizard allows you to add the field-level annotations.

This page includes the following options:

| Option | Description |
|---|---|
| **Field annotations** | |
| **TempTable** | Displays the TempTables available for the class file. Select the required TempTable. |
| **Field** | Select the field that you want to add annotations for. |
| **Annotation type** | Select the type of annotation that you want to add. The valid values are Semantic type and Foreign Key.<br>**Note:** If you want to add multiple fields to the same Foreign key, edit the annotations in the business entity file manually. |
| **Foreign Key Name** | If you selected Foreign key, provide a name for the Foreign key field. |
| **Parent Schema File** | If you selected Foreign key, browse to the file that has the schema defined.<br>**Note:** Ensure that the selected file has file-level REST or DataObject annotations and Temp-table schema. |
| **Parent Table** | If you selected Foreign key, select the table that displays the TempTables defined in the selected schema file. |
| **Parent field** | If you selected Foreign key, select a field from the parent table that can be referred to as a parent for the selected field. |
| **Finish** | The Data Object interface annotations are added to the selected ABL applications and closes the **Define Service Interface** wizard. |

# Compile files with errors

You can compile the files that have errors for a project or a folder by right-clicking the project or folder or selecting the project from the main menu and selecting the **Recompile files that have errors** option. You can also select **Recompile files that have errors** from the main menu in the **Problems** view to compile all the files with the errors.

# Export a Web UI project

You can export a Web UI project as a `.WAR` file (to a PAS for OpenEdge server) or a `.ZIP` file (to an external Web Server) to create an archive of a project that can be distributed to other systems or users.

To export the Web UI project:

1. In the Project Explorer view, select the required Web UI project that you want to export.
2. Right-click the project and click **Export** > **Web UI Application** from the context menu. The **Export Web UI Application** dialog box opens.
3. The Project drop-down list displays the project you have selected in Step 1. You can select a different project from the drop-down list.

   **Note:** The Project drop-down displays only the Web UI projects in your current workspace.

4. In the **Destination** drop-down list, specify a location or click **Browse** to select a different location for the export.
5. Select one of the following servers that you want to deploy the `.WAR` or `.ZIP` file to:
   - **PAS for OpenEdge** to export the contents of PASOEContent and Debug or Release folders to PAS for OpenEdge server.
   - **External Web Server** to export the contents of Debug or Release folders to an external Web Server.
6. Select one of the following options:
   - **Export Release Build** to export the contents of the Release folder that contains the production build used for publishing in the Kendo UI Builder structure in the Project Explorer.
   - **Export Debug Build** to export the contents of the Debug folder that contains the development build used for debugging the Kendo UI Builder structure in the Project Explorer.
7. Click **Finish**. A `.WAR` or `.ZIP` file is exported for the selected project based on build type.

## View the log files

To view the following log files, browse to the `.metadata` folder in your workspace:

- **Logs** folder — Contains feature-specific log files.
- **`.log`** file — Displays all the log entries for the Developer Studio. It includes all the log entries in the various feature-specific log files.
- **`Log4j.properties`** file — Displays the log settings for each logger. Each workspace has its own `log4j.properties` file. If you edit the properties for a specific logger, it changes the logging in the feature-specific log file and not the `.log` file. You must restart the workspace after editing the file.

**Note:** For more information on editing these log settings, see the log4j Apache documentation.

## Support for CAST function

Progress Developer Studio for OpenEdge now supports auto-completion for CAST function. For more information about the CAST function, see the *OpenEdge Development: ABL Reference* guide.

## Support for ANT tasks

OpenEdge now supports ANT tasks for ABLUnit, ABLDoc, and ABL Web App packaging on all platforms (HPUX, AIX, Solaris, Windows, UNIX, and Linux).

# RDBMS

Release 11.7.1 includes the following enhancements to the OpenEdge RDBMS:

- **Diagnostic Data Collection** — Diagnostic Data Collection allows for a variety of critical information to be saved when certain events, such as lock table overflow, BI threshold, or a system error, occur. Collecting this data can aid in the analysis of the root cause of an event. Almost all aspects of this feature are customizable, including: the events for data collection, the amount of data collected, and the formatting of the data.

  For more information, see the Web paper, *Database Diagnostic Data Collection*.

- **PROUTIL IDXCHECK Validate index against free cluster** — The Validate index against free cluster option provides a quick, deterministic check of index errors in special schemas (encryption, auditing, and key events).

## PROUTIL IDXCHECK details

Running the **Validate index against free cluster** option of PROUTIL IDXCHECK detects if index entries for the special schemas have been incorrectly deleted by PROUTIL MVSCH. These errors only exist if you executed PROUTIL MVSCH on your database with OpenEdge Releases between 11.6.0 and 11.6.3 (with no hotfix). Any 11.6.3 hotfix or a later release of OpenEdge does not cause this issue.

## Running PROUTIL IDXCHECK

To run PROUTIL IDXCHECK with the new option, use the following steps:

1. Start PROUTIL as shown:

```
proutil my-dbname -C idxcheck
```

The following menu appears:

```
Index Check Utility
===================

Select one of the following:
--------------------
All           (a/A) - Choose all the indexes
Some          (s/S) - Choose only some of the indexes
By Area       (r/R) - Choose indexes in selected areas
By Schema     (c/C) - Choose indexes by schema owners
By Table      (t/T) - Choose indexes in selected tables
--------------------
Validation    (o/O) - Change validation options
--------------------
Quit          (q/Q)

Enter your selection:
```

2. Select **Validation** by typing **o** or **O**. The following menu appears:

```
Validation Options
----------------
* 1 - Validate physical consistency of index blocks
* 2 - Validate keys for each record
  3 - Validate record for each key
* 4 - Validate key order
  5 - Validate tree
  6 - Validate index against free cluster
  R - Reset error limit, current: 500
  C - Continue to execute
  Q - Quit
```

3. Select **Validate index against free cluster** by entering **6**. You can also de-select the default options (1,2,4) by entering those numbers.
4. After selecting and de-selecting options, enter **C** to return to the main menu.
5. From the main menu select the indexes to check and execute the check.

## PROUTIL IDXCHECK output

Running the new check writes the following to the screen and log file:

```
Validate index against free cluster

Index -1610 (PUB._Constraint-Keys, _Con-Key-Ord): Validating the tree.
```

If any errors are detected, the following is written to the screen and log file:

- If the index is a non-multi-tenant or table-partitioned index:

```
Index (index number)(owner.table, index name): object not found in the area.
```

- If the index is a multi-tenant index:

```
Index (index number)(owner.table, index name) for tenant name: object not found in
the area.
```

- If the index is a table-partitioned index:

```
Index (index number)(owner.table, TP index info): object not found in the area.
```



**Warning:** If any errors are reported, contact Progress Technical Support for guidance on a recovery strategy for your database.

# Server Technologies and Tools

In Release 11.7.1, Server Technologies and Tools is updated to include the following:

## Setting JVMSTART_WAITING_TIME

If you have upgraded to Windows 10, and you get the following error while running a client to connect to a SonicMQ broker:

```
Unable to spawn SonicMQ ClientConnect process
```

Add the following environmental variable:

```
Set JVMSTART_WAITING_TIME = value_in_seconds
```