OpenEdge® Service Pack 11.7.3:

New Information

Copyright

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

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

April 2018

Last updated with new content: Release 11.7.3

Updated: 2018/04/11

Table of Contents

Preface

For details, see the following topics:

Purpose

OpenEdge® Service Pack 11.7.3: New Information documents important information about the OpenEdge Release 11.7.3 Service Pack, with references 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 11.7.3 as the most recent Service Pack for Release 11.7.

Organization

This book contains the following major sections:

- Installation and Configuration on page 17

  Contains information about new and updated third party software included in this service pack.

- Startup Parameters on page 19

  Contains information about new startup parameters.

- OpenEdge ABL on page 25

  Contains information about updates to ABL features.

- OpenEdge DataServers on page 29

  Contains information about an update to the DataServer for MS SQL Server.

- OpenEdge Management on page 31

  Contains information about updates to OpenEdge Management.

- OpenEdge RDBMS on page 33

  Contains information about updates to the RDBMS.

- OpenEdge SQL on page 39

  Contains information about password security with SQL Explorer.

- Progress Application Server for OpenEdge on page 41

  Contains information about new features in PAS for OpenEdge.

Using 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 see the OpenEdge Product Documentation Overview page on Progress Communities:

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

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\tranman`	*OpenEdge Development: Translation Manager*
`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.

Obtaining 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

Installation and Configuration

Installing the OpenEdge Release 11.7.3 Service Pack includes the following updates:

- **Infragistics** — OpenEdge uses Infragistics NetAdvantage for .NET v2016 Vol 2. The Infragistics controls are upgraded to version 16.2.20162.2182.
- **Progress Compilation Tools (PCT)** — PCT is a set of ANT libraries, designed to automate work in the OpenEdge environment. PCT helps to automate build generation in a continuous integration process for OpenEdge applications.

  With PCT, you can automate certain tasks, such as:

  - Compiling and running ABL files
  - Database related tasks
  - Running unit tests
  - Generating API documentation
  - Generating deployable artifacts

Note: Progress ships PCT with the following OpenEdge products: OpenEdge Advanced Enterprise RDBMS, OpenEdge Enterprise RDBMS, OpenEdge Personal RDBMS, OpenEdge Workgroup RDBMS, 4GL Development System, OpenEdge Application Server Basic, OpenEdge Development Server, OpenEdge Application Server Enterprise, OpenEdge Auth Gateway, Progress Dev AS for OpenEdge, Progress ProdAS for OpenEdge, Visual Translator, Query/RESULTS, OpenEdge DataServer for Oracle, OpenEdge DataServer for MS SQL, WebSpeed Workshop, Progress Developer Studio for OpenEdge, OpenEdge Studio, Client Networking.

The PCT Version included in this service pack is 208.

When installed, PCT is located in `install-dir\pct\PCT.jar`

Refer to https://github.com/Riverside-Software/pct/wiki for detailed documentation of PCT.

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.

2

Startup Parameters

The OpenEdge Release 11.7.3 Service Pack includes the following new startup parameters.

For details, see the following topics:

Catch STOP (-catchStop)

Use Catch STOP (`-catchStop`) to specify whether the AVM behaves in a new way when a STOP condition is raised. The new behavior involves being able to catch stop objects and run FINALLY blocks.

Operating system and syntax	**UNIX / Windows**	**`-catchStop` *n***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
Client Session	1	0	0	0

n

Indicates the AVM behavior when a STOP condition is raised:

- **0** — The AVM does not throw stop objects, execute relevant CATCH blocks, or execute FINALLY blocks, when a STOP condition is raised. The AVM raises and responds to STOP conditions as in prior releases. This is the default mode for version 11.7 and any of its service packs.
- **1** — The AVM throws an appropriate stop object when a STOP condition is raised, executes any appropriate CATCH block in response, and executes FINALLY blocks.

To enable the handling of stop objects in the AVM, you must recompile any existing source code that contains a CATCH block for `Progress.Lang.Error`, `Progress.Lang.SysError`, or `Progress.Lang.ProError`, and you must compile any new code with a CATCH block for a stop object. Code that references stop objects will compile successfully regardless of the value of `-catchStop`.

Once the relevant code is compiled, if you specify `-catchStop 0`, any CATCH blocks that work with stop object references are ignored by the AVM at run time.

For more information on STOP conditions and stop objects and how the AVM generates and responds to them, see the descriptions of the ON STOP phrase and the CATCH statement in the *OpenEdge Development: ABL Reference* and the description of STOP handling in *OpenEdge Development: Error Handling*.

Buffer Hash Latch Factor (-hashLatchFactor)

Use Buffer Hash Latch Factor (`-hashLatchFactor`) to tailor the number of latches allocated for the Buffer pool Hash Table (BHT) entries. The number of latches allocated is calculated based on the percent you specify.

Operating system and syntax	**UNIX / Windows**	**`-hashLatchFactor` *n***	
Use with	**Maximum value**	**Minimum value**	**Default**
Database Server	100	5	10

n

The number of BHT latches to create, specified as a percent of the BHT size, and rounded down to a whole number.

The size of the BHT defaults to approximately 25 percent of the value of `-B`. Therefore, if `-B` is set at 50000, then the buffer hash table defaults to approximately 13000 entries. If you leave `-hashLatchFactor` at the default of 10 percent, 1300 latches are allocated at startup.

Notes

- The number of latches allocated by the value of Buffer Hash Latch Factor (`-hashLatchFactor`) is fixed at startup of the primary database broker, and can not be modified while the database is online.
- If Buffer Hash Latch Factor (`-hashLatchFactor`) is specified on a connection other than the primary database broker, the value is validated as accurate and then ignored. If the validation fails, the connection is refused.
- You can set the size of the BHT with the `-hash` startup parameter.

Log file truncate time (-lgTruncateTime)

Use Log file truncate time (`-lgTruncateTime`) to specify a time to truncate the log file.

Operating system and syntax	**UNIX / Windows**	**`-lgTruncateTime` *`HH:MM`***	
Use with	**Maximum value**	**Minimum value**	**Default**
Database Server	23:59	00:00	—

`HH:MM`

> The time of day to truncate your log file, specified with a 24 hour clock. 00:00 is midnight, 23:59 is one minute before midnight. You can omit leading zeros, as long as there is a valid `H` and `M` separated by "`:`". For example 3:3 and 03:03 are equivalent.

You must specify both `-lgTruncateFrequency` and `-lgTruncateTime`. If you only specify one of the parameters, no truncation occurs. Use `-lgTruncateFrequency` to specify the day or days to truncate the log file. See Log file truncate frequency (-lgTruncateFrequency) on page 22 for more information.

When the day and time specified by `-lgTruncateFrequency` and `-lgTruncateTime` occurs, the log file is truncated, provided the database is online and no message is currently being written to the file. If the file is actively being written to, the truncation occurs as soon as the write completes. This guarantees that no message is split. If the database is offline or in single-user mode at the specified time, the database log file is **not** truncated, and there is no time-based attempt to truncate until the next time the parameters indicate.

The value for `-lgTruncateTime` can be changed while your database is online with PROMON or the _DbParams VST.

Log file truncate frequency (-lgTruncateFrequency)

Use Log file truncate frequency (`-lgTruncateFrequency`) to specify how often to truncate your log file.

Operating system and syntax	UNIX / Windows	**`-lgTruncateFrequency`** ***`day`***	
Use with	**Maximum value**	**Minimum value**	**Default**
Database Server	8	0	-1

`day`

Specify the frequency for truncating your log file. The following table describes the accepted values and their meaning.

Parameter value	Definition
0	Truncate the log file daily
1	Truncate the log file on Monday
2	Truncate the log file on Tuesday
3	Truncate the log file on Wednesday
4	Truncate the log file on Thursday
5	Truncate the log file on Friday
6	Truncate the log file on Saturday
7	Truncate the log file on Sunday
8	Truncate the log file on the last day of the month
-1[1]	Do not truncate

You must specify both `-lgTruncateFrequency` and `-lgTruncateTime`. If you only specify one of the parameters, no truncation occurs. Use `-lgTruncateTime` to specify the time of day to truncate the log file. See Log file truncate time (-lgTruncateTime) on page 21 for more information.

When the day and time specified by `-lgTruncateFrequency` and `-lgTruncateTime` occurs, the log file is truncated, provided the database is online and no message is currently being written to the file. If the file is actively being written to, the truncation occurs as soon as the write completes. This guarantees that no message is split. If the database is offline or in single-user mode at the specified time, the database log file is **not** truncated. No attempt to truncate the log file is made until the next time the day and time criteria are satisfied, while the database is in multi-user mode.

[1] You cannot specify -1 as a parameter value, but it is visible and settable in PROMON and the _DbParams VST, indicating that truncation based on frequency is disabled.

The value for the `-lgTruncateFrequency` can be changed while your database is online with PROMON or the _DbParams VST.

Log file truncate size (-lgTruncateSize)

Use Log file truncate size (`-lgTruncateSize`) to specify a size for truncating the log file.

Operating system and syntax	**UNIX / Windows**	**`-lgTruncateSize` *`n`***	
Use with	**Maximum value**	**Minimum value**	**Default**
Database Server	2147483647	0	0

`n`

A number indicating the maximum size of the database log file in MB before truncating it. 0 indicates that the database log file should not be truncated based on size.

Truncating the log file will not split a message. If the `-lgTruncateSize` value is reached while a message is being written, the log file is truncated *after* the message is completely written to guarantee that no message is split.

If the database is offline or in single-user mode when the truncate size is reached, the database is **not** truncated. The log file will be truncated the next time the database is brought online in multi-user mode, provided the size is larger than the truncate size specified at startup.

The value for the `-lgTruncateSize` can be changed while your database is online with PROMON or the _DbParams VST.

Log file archive enable (-lgArchiveEnable)

Use Log file archive enable (`-lgArchiveEnable`) to specify that the database log file should be archived before it is truncated.

Operating system and syntax	**UNIX / Windows**	**`-lgArchiveEnable`**	
Use with	**Maximum value**	**Minimum value**	**Default**
Database Server	—	—	—

If the archive file can not be created for any reason, the log file is not truncated. This ensures that no log file entries are lost.

The value for the `-lgArchiveEnable` can be changed while your database is online with PROMON or the _DbParams VST.

Log file archive directory (-lgArchiveDir)

Use Log file archive directory (`-lgArchiveDir`) to specify a directory for the archived log file.

Operating system and syntax	UNIX / Windows	**`-lgArchiveDir`** ***`dir-spec`***	
Use with	**Maximum value**	**Minimum value**	**Default**
Database Server	255 chars	—	—

`dir-spec`

A fully-qualified directory specification, indicating where archived log files are stored. The maximum size of *`dir-spec`* is 255 characters.

Log files are archived when two conditions are satisfied: log file archiving is enabled (`-lgArchiveEnable`) and either a size limit (`-lgTruncateSize`) or elapsed time (`-lgTruncateTime` and `-lgTruncateFrequency`) threshold is met.

If archiving is enabled and a directory is not specified, the archived log file is stored in directory where the database `.db` file is located.

The archive directory must exist and be writable prior to creating the log file archive, however its existence is not checked at startup. If the directory does not exist, the archive file is written to the database directory, similar to not specifying this parameter at all.

If there are any failures associated with writing the archive, such as access security or insufficient space, the archive fails. If archiving the log file fails, an error message is written to the log file, and the log file is **NOT** truncated.

Notes

- The value for the `-lgArchiveDir` can be changed while your database is online with PROMON or the _DbParams VST.
- You can archive log files from multiple databases to the same archive directory.
- Archive files use the following naming convention: *`dbname`*`.lg.`*`date-time`*`.`*`n`*

  In this filename:

  - *`dbname`* — The name of the database.
  - *`date-time`* — The date and time when the archive file is created, using the format `YYYY-MM-DD.HH-MM`.
  - *`n`* — An identifier, only incremented when the combination of *`dbname`* and *`date-time`* does not create a unique file name.

3

OpenEdge ABL

This section describes ABL enhancements in the OpenEdge Release 11.7.3 Service Pack.

For details, see the following topics:

STOP condition improvements

Release 11.7.3 includes the following ABL enhancements for STOP condition processing:

- **STOP condition handling using STOP objects is fully supported**—STOP condition handling using STOP objects was a "Technical Preview" feature in OpenEdge Release 11.7. It is now a fully implemented and supported feature for OpenEdge ABL.
- **FINALLY blocks now execute as part of STOP processing**—Prior to this release if you get a STOP condition in a block, the AVM does not run any FINALLY block unless the STOP condition is handled by either a local CATCH block or an ON STOP phrase. Now STOP conditions behave more like errors. Even if the STOP condition is not handled in the current block, the AVM runs a local FINALLY block. There are still a few, rare conditions that cause STOP to bubble up to the top of a transaction block, or out to a layer that does not access the database, and not abide by any ON STOP or relevant CATCH blocks along the way. For those conditions, FINALLY blocks along that path will not run either.

COMPILE OPTIONS phrase improvements

This release allows you to choose whether errors or warnings are displayed for COMPILE OPTIONS phrase rule violations. Prior to this release, if a violation occurred, the compiler would display the error and stop the compilation. Now, by specifying a severity level, you can control whether warnings are displayed, which do not prevent compilation, or errors, which do stop compilation. If no severity level is specified the default is `Warning`.

The following ABL syntax has been updated to specify the severity level for the COMPILE OPTIONS phrase:

```
OPTIONS options-list | OPTIONS-FILE { options-file | VALUE ( expression ) }
```

One or more rules are enforced during compilation. In the case of OPTIONS, the rules are specified by a character expression that evaluates to a comma-separated list of options. In the case of OPTIONS-FILE, the rules are specified in the same way, but supplied in a designated file.

The possible options are described in the following table:

Option	Description
`require-full-names [ :Error | :Warning ]`	All table and field names must appear as they are in the schema. The compiler’s ability to implicitly resolve abbreviated names in tables is disabled. Optionally set severity level to `Warning` or `Error`.
`require-field-qualifiers [ :Error | :Warning ]`	All buffer references (including database tables, temp-tables, and buffers) must be fully qualified. The compiler's ability to implicitly resolve the buffer to which a field reference refers is disabled. Optionally set severity level to `Warning` or `Error`.
`require-full-keywords [ :Error | :Warning ]`	All language keywords must be fully spelled out. Optionally set severity level to `Warning` or `Error`.

Note that the severity level specifies what happens when there is a failure:

- `Error` generates a message for each failure, and prevents the generation of r-code.
- `Warning` generates a message for each failure, but allows the generation of r-code, provided that the only failures during compilation are related to the option. (Other types of failure can prevent r-code generation.)
- If no severity level is specified, the default is `Warning`.

For example, the following COMPILE statement enforces full table and field names, and fully qualified references with different severity levels.

```
COMPILE myproc.p OPTIONS "require-field-qualifiers:Error,require-full-names".
```

The code below shows access of a field with a fully qualified reference, followed by unqualified and abbreviated examples of the same statement. Only the first statement compiles without error if the code is compiled using the statement above.

```
/* Accessing the field "tcfield" in the table "tbhelper" */

tbhelper.tcfield = "doUpdate".

/* The following will not compile with "require-field-qualifiers:Error"
in effect. */

tcfield = "doUpdate".

/* The following compiles with "require-full-names" with default severity level
in effect but warning messages are generated. */

tbh.tcf = "doUpdate".
```

If *`options-list`* or the contents of the options-file resolve to the empty string ("") or white space only, then no options are applied. If *`options-list`* or the contents of the options-file resolve to the Unknown value (`?`), then the compiler ignores the OPTIONS or OPTIONS-FILE phrase. If an option listed is not valid, the compile statement fails with an error. When compiling a class hierarchy, the options set in the COMPILE statement are applied to all files compiled for the hierarchy.

You can also set these options using the OPTIONS attribute of the COMPILER system handle, which can be initialized with the Compiler Options (`-compileroptionsfile`) startup parameter. (See *OpenEdge Deployment: Startup Command and Parameter Reference* for more information.) Options set using the OPTIONS or OPTIONS-FILE in the COMPILE statement override any options set in the OPTIONS attribute of the COMPILER system handle (unless *`options-list`* or the contents of the options-file resolve to the Unknown value (`?`), as described above).

Note: OPTION-FILE supports the use of the hashtag (#) to include comments in the options file. See the Compiler Options (`-compileroptionsfile`) startup parameter in *OpenEdge Deployment: Startup Command and Parameter Reference* for an example.

New SESSION EXIT-CODE

Release 11.7.3 is enhanced to include a new EXIT-CODE attribute for the SESSION system handle. The attribute can be used to optionally set an exit code, from within the application, so it's available to the operating system when the AVM process ends. This provides meaningful feedback to the shell script and allows the creation of automation scripting around the termination of the ABL application.

EXIT-CODE attribute

Returns an optional exit code set by the application, when the AVM process ends.

Data type: INTEGER

Access: Readable/Writeable

Applies to: SESSION system handle

The attribute allows the application to optionally set a program exit code. This provides meaningful feedback to the shell script and allows the creation of automation scripting around the termination of the ABL application.

This attribute is set to the Unknown value (`?`) if not set by the application.

For example:

```
SESSION:EXIT-CODE=123.
SESSION:EXIT-CODE=myIntVar.
```

Windows allows positive integer values (0-32767). UNIX restricts this to the range of 0 through 255 (inclusive). If you attempt to set an improper value for your platform, a runtime error is raised.

If an AVM process running as an AppServer or WebSpeed agent sets an `EXIT-CODE` value, a runtime error is raised.

The exit code specified by `SESSION:EXIT-CODE` can be printed from a bash shell as shown in the following example:

```
$PROEXE -p MyTest.p -b
echo $?
```

In Windows, the exit code can be printed from the command line as shown below:

```
start /wait $PROEXE -p MyTest.p -b
echo %errorlevel%
```

In Windows PowerShell the pseudo-environment variable is called by `$LastExitCode` instead of `%errorlevel%`.

Other shells in Windows may restrict the exit code to a smaller range than 0-32767. For example, Cygwin limits the value to the same range as Unix (0-255) and returns only the lower 8 bits of the value. For example, if the application sets the exit code to 300, the exit code displayed from a Windows command prompt (`cmd.exe`) is 300, while in Cygwin it is 44 (the lower 8 bits):

Integer value	Binary value
300	00000000 00000000 00000001 00101100
44	00000000 00000000 00000000 00101100

It is important to keep this in mind when choosing exit code values.

4

OpenEdge DataServers

OpenEdge Release 11.7.3 Service Pack includes the following improvement to the OpenEdge DataServer for MS SQL Server:

Performance Improvement for two level single-shot join query

Starting in this release, two level single-shot join queries are evaluated as join by server on the server side. This could result in significant performance improvements. Until now, single-shot ABL statements were evaluated on the client side.

Note:

A two level single-shot join, with a unique index used in the USE-INDEX phrase, is executed on the server side. In the case where a non-unique index is used in the USE-INDEX phrase specified with the FIRST/LAST buffer, the join is still evaluated on the client side.

5

OpenEdge Management

OpenEdge Release 11.7.3 Service Pack includes the following improvements to OpenEdge Management:

- **Removal of Customer Experience Improvement Program page** — In earlier OpenEdge releases, the **Customer Experience Improvement Program** page enabled Progress Software to collect data related to the usage trends and patterns of the product, and thereby improve the quality of the product. To eliminate collection of such data, this page has now been removed.

  This page has been removed from the OpenEdge Management initial configuration pages, and the option to enable or disable this feature has been removed from the **OpenEdge Management General Configuration** (**Options** > **General**) page.

- **Support to stop a PAS application agent** — This release provides the capability to stop a PAS application agent gracefully or forcibly. You can choose how you want to stop a PAS agent by specifying the wait time using the following options in OpenEdge Management:

  - **Wait to finish** — The time, in seconds, the agent waits to complete the requests that are currently running before stopping gracefully.

    After waiting for the specified time, if **Wait after stop** is set to 0, the agent is stopped forcibly.

  - **Wait after stop** — The time, in seconds, the agent waits after completing **Wait to finish** and before stopping forcibly.

  For more information about these options, see Progress Application Server for OpenEdge on page 41. For more information about managing PAS application agents using OpenEdge Management, see *OpenEdge Management: Progress Application Server for OpenEdge Configuration*.

- **Support to view new Database Startup Parameters** — OpenEdge Management provides support to view the following new database startup parameters:

  - **Omit log messages (`-omitLgMsgs`)**

- **Limit log file payload (`-limitLgPayload`)**
- **Log file truncate time (`-lgTruncateTime`)**
- **Log file truncate frequency (`-lgTruncateFrequency`)**
- **Log file truncate size (`-lgTruncateSize`)**
- **Buffer Hash Latch Factor (`-hashLatchFactor`)**
- **Log file archive enable (`-lgArchiveEnable`)**
- **Log file archive directory (`-lgArchiveDir`)**

For more information about the new startup parameters, see Startup Parameters on page 19. For general information about OpenEdge database startup parameters, see *OpenEdge Data Management: Database Administration*.

6

OpenEdge RDBMS

This section describes the enhancements to the RDBMS in the OpenEdge Release 11.7.3 Service Pack.

For details, see the following topics:

Buffer Hash Table concurrency

This release provides the ability to adjust the number of latches that protect the Buffer Hash Table (BHT). Each BHT latch protects a portion of the table, restricting concurrent access to that portion. With more latches, the fraction of the table protected by a single latch is reduced. Prior to this release, the number of latches was a fixed at 1024; starting in this release, you can specify the number of latches as a percent of the number of entries in the hash table. A larger number of latches should reduce contention for random access to the buffer pool. A larger number of latches will not reduce contention created by multiple users attempting to access the same data.

Specify the number of latches with the `-hashLatchFactor` startup parameter on the primary broker. If not specified, the factor defaults to ten percent (10%) of the size of the BHT. The BHT is sized at approximately one quarter (25%) of the value of `-B`. Alternately, you can set the size of the BHT with the `-hash` startup parameter. You must specify the `-hashLatchFactor` value at startup; you can not modify it while the database is online. The following table illustrates the number of latches allocated, based on specifying `-B 2000000`, and varying the `-hashLatchFactor`.

Buffers (-B)	Buffer Hash Table (approximate)	-hashLatchFactor	Number of latches (approximate)
2,000,000	500,000	5 (min value)	25,000
2,000,000	500,000	10 (default)	50,000
2,000,000	500,000	25	125,000
2,000,000	500,000	50	250,000
2,000,000	500,000	100 (max value)	500,000

For more details, see Buffer Hash Latch Factor (-hashLatchFactor) on page 20.

Log file archive and truncate

Database log files grow constantly. The ability to truncate a database log file, and optionally archive the file before truncation while the database is online, provides DBAs with the ability to save log file information, and also manage file growth.

As you configure your database to truncate and possibly archive your log file, consider the following:

- There is only one active log file, and it is named `*dbname*.lg`.
- When the log file is truncated, licensing information and the startup parameters of the primary broker are re-written to the log file.
- If truncation is configured to occur at a specific time, and the database is offline or in single-user mode, the log file is not truncated.
- If truncation is configured to occur when the log file reaches a specific size, and the database is offline or in single-user mode when the size is reached, the log file is not truncated. If the truncation size parameters are met the next time the database is brought online in multi-user mode, the log file is truncated then.
- If the database is in the middle of writing a message when the truncation size or time is reached, the log file is truncated after the message writing completes.
- You can configure your log file to be truncated by both time and size, and both are honored.
- If you choose to archive your log file, in the time between starting your database and archiving the log file, an interim log file is created and maintained in the archive directory or the database directory if no archive directory is specified.

You can manage the criteria for truncating and archiving your log file through startup parameters, the _DbParams VST, or with PROMON.

Log file truncate and archive startup parameters

Use the following startup parameters to configure truncating and archiving your database log file.

Table 1: Log file truncation parameters

Parameter	Description	For more information, see
`-lgTruncateFrequency`	Specify the frequency of when your log file is truncated. Valid values include daily, one day per week, or the last day of the month. Must be used with `-lgTruncateTime`.	Log file truncate frequency (-lgTruncateFrequency) on page 22
`-lgTruncateTime`	Specify the time of day when your log file is truncated. Valid values are based on a twenty four hour clock, 00:00 (midnight) to 23:59 (one minute before midnight). Must be used with `-lgTruncateFrequency`. **Note:** If the database is offline or in single user mode when the time and frequency parameters indicate, no truncation occurs until the next time the parameters are satisfied.	Log file truncate time (-lgTruncateTime) on page 21
`-lgTruncateSize`	Specify the size in MB when the database log file is to be truncated. Valid values range from 0 to 2147483647. The value of 0 (default) indicates that truncation based on size is disabled. **Note:** If the database is in single user mode when the log file reaches the specified size, it is not truncated. It is truncated the next time the database is brought online in multi-user mode, provided the size criterion is still met. If the log file reaches the specified size while writing a message, the database log file is truncated at the conclusion of writing the message.	Log file truncate size (-lgTruncateSize) on page 23

The table that follows describes the parameters for archiving your database log file. If you do not configure truncation of your database log file, it is not archived.

Table 2: Log file truncation parameters

Parameter	Description	For more information, see
`-lgArchiveEnable`	Specify that you want the database log file archived before truncating it. **Note:** If archiving the log file fails for any reason, the log file is not truncated.	Log file archive enable (-lgArchiveEnable) on page 23
`-lgArchiveDir`	Specify the fully qualified path name of the directory where you want the archive files written.	Log file archive directory (-lgArchiveDir) on page 24

PROMON Database Log file management

PROMON provides the ability to view and change the current settings for truncating and archiving log files. To access the **Database Log file management** from the PROMON main menu, select **R&D (Advanced options)** > **4 (Administrative Functions)** > **15 (Database Log file management)**. The following menu appears:

```
02/05/18        OpenEdge Release 11 Monitor (R&D)
15:11:14        Database log file management

        Current .lg file maintenance settings:

         1. Omit .lg file messages (-omitLgMsgs):             Not Enabled
         2. Limit .lg file payload (-limitLgPayload):         Not Enabled
         3. Archive directory (-lgArchiveDir):                Not Enabled
         4. Archive before truncation (-lgArchiveEnable):     Not Enabled
         5. Truncation frequency (-lgTruncateFrequency):      Not Enabled
         6. Maximum size before truncation (-lgTruncateSize): Not Enabled
         7. Time of day for truncation (-lgTruncateTime):     00:00


Enter a number, P, T, or X (? for help):
```

For each option, select the number to toggle the setting, or input a new value.

You can also determine the last time the log file was truncated (since startup). From the PROMON main menu, select **7 (Database Status)**.

_DbParams VST

All the startup parameters for truncating and archiving your database log file are modifiable while your database is online using the _DbParams VST.

Archiving

When you enable archiving of your truncated log file, the following process occurs:

1. An interim archive file is created in your archive directory, or in the database directory, if an archive directory is not specified. The interim archive file is named,`dbname.lg.date-time.n.tmplg`, where the `date-time` is the time the temporary archive is created.
2. Over time, the database broker writes log file entries to the interim archive in 8K chunks.
3. When the database log file is truncated, the broker writes any remaining log messages to the interim archive.
4. The interim archive is re-named to the archive file name, using the `date-time` when the truncation and archive criteria are met. The file is named, `dbname.lg.date-time.n`.
5. A new temporary archive is created the next time the broker has an 8K chunk to write to it.
6. Repeat Steps 2-5.

The temporary archive is deleted at database shutdown, or if the database log file is truncated by PROLOG.

Archive file naming convention

When you archive your log file, it is given a name using the convention, `dbname.lg.date-time.n`.

In this file name:

- `dbname` — The name of the database.
- `date-time` — The date and time when the archive file is created, using the format `YYYY-MM-DD.HH-MM`.
- `n` — An identifier, only incremented when the combination of `dbname` and `date-time` does not create a unique file name.

The interim archive file uses the same convention, but adds a `.tmplg` extension to the file name.

Truncated and archived files

As you truncate and archive your log files, the log and archive files are modified as follows:

- **Truncated log file** — When you truncate your log file, all existing information in your log file is deleted, and the log file is re-created. Near the top of your log file is the following message:

  ```
  (19219) The database log file was truncated.
  ```

  Your log file is also re-loaded with all your license information, and the primary broker startup parameters with their current values (which could have been changed while the database is online).

- **Interim archive file** — When you specify that you want to archive your log file, an interim archive is created and filled, in anticipation of being archived. When you enable archiving, the interim log file is named,`dbname.lg.date-time.n.tmplg`, where the `date-time` is initially the start time, and over time is the time of the previous archive. The interim log file is renamed when archiving occurs, and a new interim log file is created. The interim log file is deleted when the database is shut down or the log file is truncated with PROLOG.

- **Truncated and archived log file** — When you truncate and archive your log file, the log file is first archived, then all existing information in your log file is deleted, and the log file is re-created. The following messages are written to your truncated log file:

  ```
  The database log file was archived to archive-dir/archive-file.
  (19219) The database log file was truncated.
  ```

In this instance, *`archive-dir`* is the fully qualified directory name where the log file is created, and *`archive-file`* is the name of the log file archive.

Your log file is also re-loaded with all your license information, and the broker startup parameters with their current values (which could have been changed while the database is online).

- **Archived log file** — When your archive log file is created, near the end of the archive, are the following lines (written to the log file when preparing to archive):

```
(19233) Preparing to truncate the database .lg file.
(19232) Database .lg file archive in process.
```

7

OpenEdge SQL

The OpenEdge Release 11.7.3 Service Pack includes the following update for OpenEdge SQL:

Securely specifying a password in SQL Explorer

The purpose of SQL Explorer password security is to find a safe and secure way to provide password security.

While using SQL Explorer, the password is specified using the `-password` option which is in clear text and is visible. SQL Explorer password security helps in blocking this security breach. Using this feature, the password value cannot be viewed by UNIX operating system tools like `ps` and any other Windows equivalents.

For SQL Explorer password security, the options are (in the sequence of most secured to least):

- **Prompt for password** — Previously, the `-password` option of SQL Explorer was mandatory when it was used along with `-user` option. Now this is optional. The SQL Explorer utility prompts for password when it is not provided in the command line. While typing, the password is neither visible in the console and nor viewed by any operating system utility.
- **Using sqlexp in batch mode** — SQL user can redirect the output of `echo` / `cat` commands to `sqlexp` via ‘pipe’ (‘|’). This option is less secure than the one above.
- **Using genpassword utility** — This option is for those who want to use the command line to provide the password. Use the `genpassword` utility to generate an encrypted password instead of using the actual password. Then use the encrypted password in the SQL Explorer tool.

  If the password is `test` for user `userA` then use the `genpassword` utility to get an `OECH1` encrypted password for `test`, as shown:

  ```
  genpassword -password test
  encrypted-password
  ```

You can take the *`encrypted-password`* output from the `genpassword` utility and prefix with `oech1::`, and supply that to `-password` option of `sqlexp`.

For more information on the `genpassword` utility, see *OpenEdge Getting Started: Installation and Configuration*.

8

Progress Application Server for OpenEdge

This section describes the new features in PAS for OpenEdge that are included in the OpenEdge Release 11.7.3 Service Pack.

For details, see the following topics:

Moving Applications to PAS for OpenEdge

In support of the new `paspropconv` utility, described in the Using PASPROPCONV for Migration on page 44, the following content is available:

- Quick Start: Moving Your Classic AppServer Applications to the Progress Application Server for OpenEdge is a step by step guide for moving an application to a PAS for OpenEdge test instance.
- Moving Your Classic AppServer Applications to the Progress Application Server for OpenEdge is a companion video demonstrating moving a sample ABL application to PAS for OpenEdge using the APSV transport.

It is expected that additional changes are required before moving applications to a live production system. Please review the following documentation to learn more about those changes:

- *Progress Application Server for OpenEdge: Application Migration and Development Guide*
- *Progress Application Server for OpenEdge Configuration*
- *Progress Application Server for OpenEdge Administration Guide*
- *Progress Application Server for OpenEdge Tuning Guide*

REST API Reference for Agent Management

OpenEdge 11.7.3 introduces the following two changes to how you manage a multi-session agent in a PAS for OpenEdge instance using REST APIs:

- **Stop a multi-session agent** — Prior to 11.7.3, the REST API for stopping an agent would stop the agent immediately, even if the agent were currently serving a request. Now, you have two optional parameters, `waitToFinish` and `waitAfterStop`, which you can use to let an agent finish processing a request before it is stopped.
- **Add a multi-session agent** — Now, there is a REST API that you can use to add an agent to an ABL application.

Stop a multi-session agent

You can use the `waitToFinish` parameter to specify, in milliseconds, how long the stop process must wait before stopping an agent, if the agent is currently serving a request. `waitToFinish` stops new requests to the agent and waits for the current request to complete. If the current request does not complete in the specified time, a STOP message is sent to the agent.

You can also use the `waitAfterStop` parameter to specify an additional wait time, in milliseconds. `waitAfterStop` checks if the agent has indeed stopped. If the agent has not stopped, then a KILL message is sent to the agent at the end of the specified time. In either case, if the agent finishes processing its current request before `waitToFinish` expires, the agent will immediately shut down.

Note: Both `waitToFinish` and `waitAfterStop` are optional parameters. If you do not specify these, the agent will be stopped immediately, irrespective of whether the agent is currently idle or serving a request.

HTTP Operation

```
DELETE
```

URI

```
//host_name:port/oemanager/applications/App_name/agents/agentID?
waitToFinish=time-in-milliseconds&waitAfterStop=time-in-milliseconds
```

Media type

```
application/vnd.progress+json
```

Response codes

```
200 Success
403 Access Denied
500 Unexpected Server Error
```

Command-line example

```
curl -X DELETE -v
http://localhost:16680/oemanager/applications/oepas1/agents/AG-sCIVXeFSQYmcb7RcHo10Zw?waitToFinish=
60000 -u tomcat:tomcat -H "Content-Type: application/vnd.progress+json"
```

Request body example

NA

Response body example

```
{
"result": {
"agentID": "AG-sCIVXeFSQYmcb7RcHo10Zw"
},
"versionStr": "v11.7.3 ( 2018-02-12 )",
"versionNo": 1,
"outcome":"SUCCESS",
"errmsg":"",
"operation": "STOP AGENT ",
}
```

Add a multi-session agent

Add a multi-session agent to an ABL application.

HTTP Operation

`POST`

URI

```
//host_name:port/oemanager/applications/App_name/addAgent
```

Media type

```
application/vnd.progress+json
```

Response codes

```
200 Success
403 Access Denied
500 Unexpected Server Error
```

Command-line example

```
curl -X POST -v http://localhost:16680/oemanager/applications/oepas1/addAgent
```

Request body example

NA

Response body example

```
{
"result": {
"agentID": "9RT0PpxRTk6qPS1XEA_QSg", "pid":"84708","state":"AVAILABLE"
},
"versionStr": "v11.7.3 ( 2018-02-12 )",
"versionNo": 1,
"outcome":"SUCCESS",
"errmsg":"",
"operation": "GET ABL OBJECTS REPORT",
}
```

Using PASPROPCONV for Migration

The `paspropconv` tool or utility is introduced in OpenEdge 11.7.3. It is useful for migrating application logic from a Classic AppServer to a Progress Application Server for OpenEdge (PAS for OpenEdge) instance. The tool takes as input the properties of a Classic AppServer from either a `ubroker.properties` file or a `.merge` file (created using the `mergeprop` utility) and generates several files, including:

- **`pasoename.ubrokername.oemerge`** — This file contains two sections. The **first section** explains all the conversions made while adapting the Classic AppServer properties to PAS for OpenEdge. It also indicates properties that were not adapted, properties that you should review and modify, and other important information. Please take time to read this section. The **second section** can be used to add the new values to your PAS for OpenEdge instance using the `oeprop` utility.
- **`pasoeAppName_setenv.sh`** and **`pasoeAppName_setenv.bat`** — These environment files are generated if there is a corresponding `[Environment.ubrokerName]` section in the `ubroker.properties` or the `.merge` file. Copy the platform-specific file to the bin directory of the instance to have the environment variables read at the time your PAS for OpenEdge instance starts.
- **`paspropconv.log`** — Review this file for any errors that may have occurred during the conversion process.
- **`paspropconv_notesdb.en`** — This file is used to create the comments section of the `pasoename.ubrokername.oemerge` file. Presently, it is available only in English.

Note: The `paspropconv` utility requires `$DLC/bin/mergeprop.bat` to work. Therefore, it is bundled with the Classic AppServer in OpenEdge 11.7.3. If you are using an earlier version of the Classic AppServer, use the `mergeprop -listall` command to create a `.merge` file. Then, you can use the `.merge` file for conversion using `paspropconv`.

For more information, see *Progress Application Server for OpenEdge: Quick Start Guide*.

Syntax

```
paspropconv {--ubrokerPropsFile absolute-path-of-ubroker.properties --ubrokerName
 UBroker.AS.ubrokername |
      -ubrokerMergeFile name-of-the-ubroker-merge-file}
            --pasoeAppName ABL-application-name
  [--pasoeWebAppName ABL-web-app-name]
            [--pasoeMergeFile pasoe-merge-filename]
  [pasoeSetEnvFile name-of-setting-evn-variables-file]
  [convNotesDBFile name-of-the-file-containing-conversion-notes]
  [--logFile log-filename]
            [--loggingLevel logging-level]
  [--logAppend 0-or-1]
            [mergepropExe name-of-mergeprop-utility]
```

Parameters

`--ubrokerPropsFile` *`absolute-path-of-ubroker.properties`*

Specify the absolute path of the `ubroker.properties` file. This parameter is required, unless you use `ubrokerMergeFile`.

`--ubrokerName` *`UBroker.AS.ubrokername`*

Specify the fully qualified name of the AppServer broker (`UBroker.AS.brokername`).

`--ubrokerMergeFile` *`name-of-the-ubroker-merge-file`*

Specify the absolute path of the `ubrokerMergeFile` that is generated using `mergeprop`. This parameter is required, unless you use `ubrokerPropsFile`.

`--pasoeAppName` *`ABL-application-name`*

Specify the name of the ABL application to which you want to migrate your application logic. If you want to migrate the application logic to the default ABL application, specify the name of your PAS for OpenEdge instance.

`--pasoeWebAppName` *`ABL-web-app-name`*

Specify the name of the ABL web application, inside the ABL application that you already specified, to which you want to migrate your application logic. By default, the application logic is migrated to the ROOT ABL web application.

`--pasoeMergeFile` *`pasoe-merge-filename`*

Specify the name of the file that is one of the expected file outputs of running the `paspropconv` tool. You will subsequently merge the contents of this file with your PAS for OpenEdge instance's openedge.properties file. By default, the file is named as `pasoeAppName.ubrokername.oemerge`.

`--pasoeSetEnvFile` *`name-of-setting-evn-variables-file`*

Specify the name of the file that you will later copy to the PAS for OpenEdge instance's bin folder, so that environment variables are properly set when the instance is started or restarted. This file is generated on running `paspropconv`. By default, the generated files are named as `pasoeAppName_setenv.sh/.bat` (for UNIX/Windows).

`--convNotesDBFile` *`name-of-the-file-containing-conversion-notes`*

Specify the name of the file. This file is generated on running `paspropconv`. By default, the generated file is named as `paspropconv_notesdb.en`.

`--logFile` *`log-filename`*

Specify the name of the `paspropconv` tool's log. By default, the log is named `paspropconv.log`.

`--loggingLevel` *`logging-level`*

This parameter determines how much information is captured in the tool's log. Possible values are `0`, `1`, `2`, `3`, `4`, and `5`. By default, the parameter is set to `2`.

`--logAppend` *`0-or-1`*

This parameter determines whether information captured for every execution of `paspropconv` replaces the existing information in the log or appends to the end of the log. If you want information replaced, specify `0`. If you want information appended, specify `1`, which is the default value.

`--mergepropExe` *`name-of-mergeprop-utility`*

Specify the full pathname of the `mergeprop` utility, if you are not using the default location (*`install-dir`*`/bin/mergeprop.bat`). The default value is `mergeprop`.

`--help`

If you need help with the `paspropconv` tool, run `paspropconv --help` to see the complete list of parameters available with the tool.

Note: You must prefix each parameter listed above with a double dash (**--**), otherwise the command will not work.

Deploying an ABL Web Application Using TCMAN DEPLOY

OpenEdge 11.7.3 introduces an additional parameter, `-l`, to the `tcman deploy` command.

Use `-l` to deploy an ABL web application to a PAS for OpenEdge instance without having to restart the instance. The parameter loads the context that is necessary for the successful online deployment of a web application.

The revised syntax is provided below.

Syntax

```
tcman.sh deploy general_options -l -u user_id:password -a app_name war_file_path
 abl_app_name
```

Note: The Tomcat manager (`oemanager.war`) must already be deployed for the `-l` switch to work. In the PAS for OpenEdge development server, the Tomcat manager is by default deployed in the default instance, `oepas1`. For all new instances in both the development and the production servers, you have to deploy the Tomcat manager first before you can use this switch. Also note that the `-l` switch can be used only in conjunction with the `-u` parameter.

Example

Deploy and rename `oeabl.war` (a web application that implements OpenEdge adapters) to the `acme1` instance of the core `pashome` server online:

```
/psc/acme1/bin/tcman.sh deploy -l -u tomcat:tomcat -a oeadapters
/psc/pashome/extras/oeabl.war
         OK - deployed /psc/pashome/extras/oeabl.war to local directory
/psc/acme1/webapps
```

Note: The `$CATALINA_HOME/extras` directory (`/psc/pashome/extras` in the example above) also contains number of instance management applications, including `host-manager.war`, `manager.war`, and `oemanager.war`.

Support for configuring JWT field name for scope

PAS for OpenEdge release 11.7.3 supports configuration of JWT field name for scope as part of its support for authentication with OAuth2 and JWT.

When mapping Self-contained JWT fields to OpenEdge Client-Principal attributes, a JWT has a recommended field name `scope` to hold the scope of the authenticated user. However, you can configure this field name, as shown below, using the Authorization Server that issues it:

```
jwtToken.scopeNameField={ scope }
```

If the configured field name for scope is not available in the JWT, then the JWT uses `PSCUser` as the default scope. You can set this default scope using the `jwtToken.defaultRoles` property in the `oeablSecurity.properties` file.

Refer to JWT issuer's documentation to find which field name contains the scope of the authenticated user and map its claim to the scope (Role).