Progress®



OpenEdge DevOps Framework

OpenEdge®

Copyright

© 2021 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

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

DataDirect Cloud, DataDirect Connect, DataDirect Connect64, DataDirect XML Converters, DataDirect XQuery, DataRPM, Defrag This, Deliver More Than Expected, DevReach (and design), Icenium, Inspec, Ipswitch, iMacros, Kendo UI, Kinvey, MessageWay, MOVEit, NativeChat, NativeScript, OpenEdge, Powered by Chef, Powered by Progress, Progress, Progress Software Developers Network, SequeLink, Sitefinity (and Design), Sitefinity, Sitefinity (and design), SpeedScript, Stylus Studio, Stylized Design (Arrow/3D Box logo), Styleized Design (C Chef logo), Stylized Design of Samurai, TeamPulse, Telerik, Telerik (and design), Test Studio, WebSpeed, WhatsConfigured, WhatsConnected, WhatsUp, and WS_FTP 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, Chef Automate, Chef Compliance, Chef Desktop, Chef Habitat, Chef WorkStation, Corticon.js, Corticon Rules, Data Access, DataDirect Autonomous REST Connector, DataDirect Spy, DevCraft, Fiddler, Fiddler Everywhere, FiddlerCap, FiddlerCore, FiddlerScript, Hybrid Data Pipeline, iMail, JustAssembly, JustDecompile, JustMock, KendoReact, NativeScript Sidekick, OpenAccess, PASOE, Pro2, ProDataSet, Progress Results, Progress Software, ProVision, PSE Pro, Push Jobs, SafeSpaceVR, Sitefinity Cloud, Sitefinity CMS, Sitefinity Digital Experience Cloud, Sitefinity Feather, Sitefinity Insight, Sitefinity Thunder, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, Supermarket, SupportLink, Unite UX, 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.

July 2021

Product version: Progress OpenEdge 12.4

Updated: 2021/06/29

Table of Contents

Part I: Introduction....7

Learn About the OpenEdge DevOps Framework....7

Part II: Gradle plugins....9

Prerequisites....10
Gradle....10
Plugins....11
Default tasks....12
Sample propath file....12
build.config file....12
Sample build.config file....18
Custom tasks....18
ABLExtension....19
ABLCompile....21
DBConnection....27
ABLRun....29
ABLUnit....33
PL....34
BackupDB....35
CreateDB....36
LoadDBSchema....38
Sample custom task....40
Migrate from Apache Ant....42

Introduction

For details, see the following topics:

- Learn About the OpenEdge DevOps Framework

Learn About the OpenEdge DevOps Framework

Read the End User License Agreement.

Mission critical business applications go through many cycles of incremental changes over the course of their lifespan, complete with code changes, compilation, builds, and validation. Often these cycles are repeated multiple times a day, before a release is made available to users. Continuous integration (CI) is a practice that focuses on optimizing the process of generating these incremental builds and validating them. For ABL applications, the OpenEdge DevOps Framework is designed to help with implementing an efficient CI pipeline that handles compilation, repository integration, testing, and packaging.

The OpenEdge DevOps Framework is comprised of a set of plugins designed to address the requirements of two types of users:

1. Users who are new to the CI process and want a simple way to set up their pipeline.
2. Advanced DevOps engineers with a complex CI process and additional flexibility needs.

The OpenEdge DevOps Framework also provides the convenience of sharing the CI pipeline configuration between the development and production build processes.

The OpenEdge DevOps Framework comes with two variations of Gradle plugins. The following sections will cover prerequisites for using the OpenEdge DevOps Framework and some basic concepts of Gradle before getting into the details of the plugins.

II

Gradle plugins

For details, see the following topics:

- Prerequisites
- Gradle
- Plugins
- Default tasks
- Sample propath file
- build.config file
- Sample build.config file
- Custom tasks
- Migrate from Apache Ant

Prerequisites

Requirements:

- Certified for OpenEdge 12.2 and later
- ABL installed with OpenEdge 12.2 and later
- Gradle 5.6.x
- Java 11
- PCT, which is used from DLC/pct
- An ABL project
- A basic understanding of Gradle

The following platforms are supported for the OpenEdge DevOps Framework:

- Windows
- Linux
- AIX
- Solaris

Gradle

Gradle is an open-source project automation tool that is an evolution of the concepts used in Apache Ant and Apache Maven. Instead of traditional XML, Gradle uses a domain-specific language based on Groovy for project configuration. Gradle intelligently determines which parts of a build tree are up to date, and only executes the parts of the build tree that are needed. This process allows builds to be smarter, faster, and more efficient because it eliminates redundancies in the build phase. The OpenEdge DevOps Framework can be used with Gradle, and therefore some of the OpenEdge DevOps Framework plugins follow Gradle conventions.

Some of these conventions are outlined in this topic. For more information about Gradle, see the Gradle documentation.

Basic concepts

The following are basic Gradle concepts and components that will be helpful when you use the OpenEdge DevOps Framework:

- Plugin: Gradle plugins add tasks objects and conventions to your Gradle environment. Plugins are the primary method by which you extend your capabilities using Gradle.
- Task: A Gradle task is the smallest piece of work in a Gradle project. A task is the building block of Gradle functionality.
- Task type: Enhanced tasks in Gradle contain built-in behaviors, rather than executing existing ones in the Gradle environment. The OpenEdge DevOps Framework makes use of these task types, and provides users with the ability to define their own tasks.
- Task dependencies: Gradle tasks can depend on other tasks, and these dependencies do not require the tasks to be defined in order.
- Groovy DSL: Gradle uses Groovy as a build language.

For more information about these and other Gradle components, see the Gradle documentation.

Plugins

The OpenEdge DevOps Framework comes with two plugins: `progress.openedge.abl` and `progress.openedge.abl-base`.

progress.openedge.abl-base

This plugin defines all capabilities of the OpenEdge DevOps Framework. The plugin capabilities are defined under extensions and task types. Extensions are global configurations available to all the task types. Task types define individual task capabilities like compile and database connection. This plugin is intended for users who have ABL source files that are not structured as Progress Developer Studio projects. Users should define tasks for each task type in order to use task types with tasks. For more information about how to use this plugin, see Custom tasks.

progress.openedge.abl

This plugin depends on the Progress Developer Studio project structure and is recommended for Progress Developer Studio users. With this plugin, these users do not need to maintain separate configuration for development environments and build environments. Users that prefer using predefined tasks can also use this plugin provided the users add and configure the required `build.config` and `.propath` files for their source artifacts. Tasks in this plugin are predefined, and the necessary parameters required for compiling ABL files can be configured in the `build.config` file that is part of each Developer Studio project. For more information about the `build.config`, see OpenEdge DevoOps Framework configuration file and OpenEdge DevOps Framework configuration parameters.

Default tasks

The following tasks are predefined in the `progress.openedge.abl` plugin and can be used with the `build.config` file to build ABL projects. Any required configuration must be done in the `build.config` file:

- `build`: Builds the project. This task compiles the project and generates the r-code. This task relies on the `compileAbl` task.
- `compileAbl`: Compiles an ABL project. This task relies on the following src entry compile tasks:
  - `dbConnections`: Holds ABL database connection details that are defined in the config file.
  - `generateCompileAblResultJson`: Generates a unified result of the JSON output of different tasks. This is used if the property `outputType` is set in `compileOptions`.
- `clean`: Cleans the project by removing the build output, and then compiles the project again in a clean environment.

Sample propath file

The following is a sample propath file:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<propath version="12.3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="schema/propath.xsd">
    <propathentry env="all" kind="src" path="@{ROOT}\src" platform="All"/>
    <propathentry env="gui" kind="con" path="com.openedge.pdt.text.GUI_LIBRARIES"
platform="All"/>
    <propathentry env="tty" kind="con" path="com.openedge.pdt.text.TTY_LIBRARIES"
platform="All"/>
    <propathentry env="all" kind="con" path="com.openedge.pdt.text.DLC_PATHS"
platform="All"/>
   <propathentry env="all" kind="lib" path="@{ROOT}\lib\sports2000.pl" platform="All"/>
</propath>
```

This plugin can be configured to provide a custom propath file path by providing the variable `CUSTOM_PROPATH_FILE`. This variable can be configured in the following ways:

1. Set as a system property (`-DCUSTOM_PROPATH_FILE=`*`propath-file-path`* set on the command line)
2. Set as a Gradle property (`CUSTOM_PROPATH_FILE=`*`propath-file-path`* set in the `gradle.properties` file)
3. Set as an environment variable

build.config file

The `build.config` file contains all the project configurations required for you to build a project. The `build.config` file allows you to move your project's property settings along with your project so that your development and integration environment uses the same configuration. You do not need to set up the configurations again.

This topic describes the parameters in the `build.config` file.

Note: Progress recommends that you do not modify the `build.config` file manually within Progress Developer Studio. All the parameters can be updated from the Progress Developer Studio user interface (**Project Properties**) with the exception of the following:

- `streamIO.enabled`
- `languages.list`
- `textSegGrow`
- `minSize.enabled`

You can update the `build.config` file manually to build the project outside of Progress Developer Studio.

The `database` and `compilableFileExtensions` parameters are not used with Progress Developer Studio.

For file portability, Progress recommends that you use variables instead of an absolute path for the following attributes:

- `'buildDir'`
- `'oeide.xrefXmlDir'`
- `'oeide.preCompileCallbackRoutine'`
- `'avm.wrkDir'`
- `'avm.avmOptions.tmpDir'`
- `'avm.avmOptions.assembliesDir'`

The following default variables are supported:

- **`ROOT`**—Project's root location
- **`WRKDIR`**—Working directory; set during installation

Note: The **system properties**, **system environment variables**, and **Eclipse variables** are also supported.

The variable should be used as `"${`*`variable-name`*`}"`, for example, `buildDir="${ROOT}/myBuildDir"`

Global parameters

Parameters	Default value	Description
`buildDir=''`	—	Specifies the project build directory. This directory contains the saved r-code files. If this field is blank, then the r-code files are saved in the same directory as the source files.

oeide parameters

The following table contains the IDE-specific configurations that are only used by Progress Developer Studio for OpenEdge.

Parameter	Default value	Description
`useSharedAVM=''`	`false`	This parameter specifies whether to use a shared ABL Virtual Machine (AVM) when building projects in Progress Developer Studio. This parameter uses a Boolean value to specify whether the project uses a shared or a dedicated AVM.
`oeideEvents=''`	`true`	This parameter uses a Boolean value to enable or disable oeide events. This parameter enables you to write procedures that use the ABL SUBSCRIBE statement to capture and respond to those events.
`useGlobalToolboxVD=''`	`false`	This parameter uses a Boolean value to enable or disable the global Visual Designer toolbox.
`addDefaultParams=''`	`true`	This paramter specifies whether to use the default startup parameters when starting the AVM. This parameter uses a Boolean value to enable or disable the passing of default AVM parameters.
`hideTTYConsole=''`	`true`	This parameter uses a Boolean value to enable or disable the project-specific run-time console.
`useProjectCompilerSettings=''`	`false`	This parameter uses a Boolean value to enable or disable the use of project-specific `strictOptions` under `compile`.
`saveRCode=''`	`true`	This parameter uses a Boolean value to enable or disable persistant r-code on compiling sources.
`xrefXmlDir=''`	—	This parameter specifies the output value of the `XREF-XML` option when `xrefXml.enabled='true'`. This parameter uses a string value to specify the path where cross-reference information is saved in an XML file.

Parameter	Default value	Description
`preCompileCallbackRoutine=''`	—	This parameter uses a string value to specify the path to an ABL file that is run before compilation. For example, this option can used to set `SECURITY-POLICY:XCODE-SESSION-KEY`.

AVM parameters

Parameter	Default value	Description
`wrkDir=''`	The project's root directory	This parameter uses a string value to specify the working directory in which the AVM starts.

AVM option parameters

The following table lists the AVM option parameters (defined in the `build.config` file as `avmOptions`).

Parameter	Default value	Description
`tmpDir=''`	The DLC working directory	This parameter specifies the temporary directory for the AVM run time (equivalent to the `-T` startup parameter option). This parameter uses a string value to specify the path.
`tty.enabled=''`	false	Flag to use `_progres` or `prowin` executables. For `_progres`, the value is `true`; otherwise, it is `false`. This parameter uses a Boolean value.
`startupParameters=''`	—	This parameter uses a string value to specify the startup parameters as a string.
`assembliesDir=''`	—	Specifies the location of the `Assemblies` directory (equivalent to the `-assemblies` startup parameter option). This parameter uses a string value.

Compile parameters

Parameter	Default value	Description
compilableFileExtensions=''	p, w, cls, pgen, html, htm	This parameter allows you to set the compilable file extensions.

Compile option parameters

The following table lists the Compile option parameters (defined in the build.config file as compileOptions).

Parameter	Default value	Description
multiCompile.enabled=''	false	To enable the COMPILER:MULTI-COMPILE attribute, set the value to true.
xcodeKey=''	—	XCODE option in the COMPILE statement. Specify the expression as a string.
xrefXml.enabled=''	false	XREF-XML option in the COMPILE statement. To enable this option, set the value to true. Specify the expression as a Boolean value.
streamIO.enabled=''	false	STREAM-IO option in the COMPILE statement. To enable this option, set the value to true. Specify the expression as a Boolean value.
minSize.enabled=''	false	MIN-SIZE option in the COMPILE statement. To enable this option, set the value to true. Specify the expression as a Boolean value.
attrSpace.enabled=''	false	To enable the ATTR-SPACE option in the COMPILE statement, set the value to true.

Strict option parameters

The following table lists the Strict option parameters (defined in the build.config file as strictOptions).

Parameter	Default value	Description
requireFullNames=''	ignore	This parameter specifies the OPTIONS require-full-names in the COMPILE statement. Value can be ignore, warning, or error.
requireFieldQualifiers=''	ignore	This parameter specifies the OPTIONS require-field-qualifiers in the COMPILE statement. Value can be ignore, warning, or error.
requireFullKeywords=''	ignore	This parameter specifies the OPTIONS require-full-keywords in the COMPILE statement. Value can be ignore, warning, or error.
requireReturnValues=''	ignore	This parameter specifies the OPTIONS require-return-values in the COMPILE statement. Value can be ignore, warning, or error.

Languages parameters

Parameter	Default value	Description
list=''	—	Comma-separated list of language segments to include in the compiled r-code.
textSegGrow=''	—	This parameter specifies the TEXT-SEG-GROW option. The value must be an integer. Supported only when a language list is provided.

Sample build.config file

The following is a sample `build.config` file.

```
buildDir='build2'

oeide{
    //used by Developer Studio
}

avm{
    wrkDir=''
    avmOptions {
        //same as attributes supported for avmOptions in ABLCompile task
        tmpDir=''
    }
    database {
        con1{
            //same as attributes supported in DBConnection task
            dbName =
        }
        con2{
            //same as attributes supported in DBConnection task
            connectionParameters='-S 8000'
        }
        //note con1 and con2 are name of the DBConnection task created and should be
unique
    }
}

compile{
    //Compilable file extensions, defaults to "p,w,cls,pgen,html,htm"
    compilableFileExtensions=''
    compileOptions{
        //same as attributes supported for compileOptions in ABLCompile task
    }
}
```

Custom tasks

You can create a custom Gradle tasks with the OpenEdge DevOps Framework using the `progress.openedge.abl-base` plugin.

Custom tasks

When writing custom Gradle tasks for your ABL project, you can use task types. Task types allow the tasks of the same type to use the same configurations. The following task types are provided in the `progress.openedge.abl-base` plugin for use with custom tasks:

- `ABLCompile`: Task to compile ABL code. For more information, see ABLCompile.
- `DBConnection`: Task for database connection. For more information, see DBConnection.

Note: Tools like Jenkins may not function correctly when using the Gradle command `progradle`. If you are using Jenkins or a similar tool you may want to use a gradle wrapper script or identify your Gradle installation location within the tool.

ABLExtension

This extension is a set of ABL language-specific configurations that are available to all task types.

Methods

Method	Description	Example
`propath`	Sets the AVM's PROPATH.	`propath('path1', "${dlcHome}/path2")`
`avmOptions(Closure avmOptions)`	AVM options. The argument should be provided as a closure. See `avmOptions` in the ABLCompile section for more information.	`avmOptions{ tmpDir="path" tty.enabled=true }`
`dbConnection({})`	Adds database configurations. See the ABLCompile section for more information.	`dbConnection{ dbName = 'sports2000' connectionParameters='-S 8000'}`
`dbConnectionReferenceId(String... refIds)`	Adds database task references. See the *id* property of DBConnection for more information.	`dbConnectionReferenceId("id1", "id2")`
`compileOptions(Closure compileOptions)`	Compile options. The argument should be provided as a closure. See `compileOptions` in the ABLCompile section for more information.	`compileOptions{ multiCompile.enabled=true outputType="json"}`

Properties

Property	Required?	Description	Example	Default value
`dlcHome`	Yes	The OpenEdge DLC path. This property can be set in the following ways: **1.** Set *DLC* as a system property (`-DDLC=<dlCPath>` set on the command line) **2.** Set *DLC* as a Gradle property (`DLC=<dlCPath>` set in `gradle.properties` file) **3.** Set *DLC* as an environment variable The order of priority is 1 > 2 > 3.	None	None
`openedgeVersion`	No (read-only property)	Specifies the OpenEdge version. This can be used for logging. This is a read-only property.	None	None
`rcodeDir`	No	Sets the directory where the r-code will be generated.	*rcodeDir="rcode2" or rcodeDir="path"*	*${buildDir}/rcode*
`propath`	No	Sets the AVM's PROPATH.	*propath* = files('path1', "${dlcHome}/path2")	None
`wrkDir`	No	Sets the working directory.	*wrkDir* = 'path1'	"${*projectDir*}".
`avmOptions{}`	No	AVM options. Use the method `avmOptions(Closure closure)` to set the properties.	None	None

Property	Required?	Description	Example	Default value
`dbConnections`	No	Database connection details. To add a database connection, use methods, `dbConnection(Closure connection)` or `dbConnectionReferenceId(String... refIds)`.	None	None
`compileOptions{}`	No	Compile options. See the ABLCompile section for more information.	None	None

Sample code snippet

The following code snippet is an example using ABLExtension:

```
abl{
    rcodeDir='${buildDir}/myRcode'
    wrkDir = "."
    propath("${dlcHome}/tty/")
    avmOptions{
        tmpDir="${buildDir}"
        tty.enabled=true
    }
    compileOptions{
        multiCompile.enabled=true
        outputType="json"
    }
    dbConnection{
        dbName = 'sports2000'
        connectionParameters='-S 8000'
    }
    dbConnectionReferenceId("id1", "id2")
}
```

ABLCompile

The ABLCompile task compiles ABL code.

Methods

Method	Description	Example
`exclude(String... excludes)`	Adds an ANT-style exclude pattern to this task.	`exclude('**/*.txt', '**/*.conf')`
`include(String... includes)`	Adds an ANT-style include pattern to this task.	`include('**/*.p', '**/*.cls')`

Method	Description	Example
`source(String... sources)`	Adds a source to this task after the include and exclude patterns are applied. Sources can have an absolute or relative path to the root directory.	`source('src1', 'src2')`
`propath(String... propaths)`	Sets the AVM's PROPATH. This method will be appended to the PROPATHs provided in ABLExtension	*`propath`*`('path1', "${dlcHome}/path2")`
`avmOptions(Closure avmOptions)`	AVM options. The argument should be provided as a closure. See `avmOptions` in the ABLCompile section for more information.	`avmOptions{ tmpDir="path" tty.enabled=true}`
`dbConnection(Closure connection)`	Adds database configurations to this task.	`dbConnection{ dbName = 'sports2000' connectionParameters='-S 8000'}`
`dbConnectionReferenceId(String... refIds)`	Adds database task references. See the *id* property of DBConnection for more information. You must add the database tasks' dependency to this task.	`dbConnectionReferenceId("id1", "id2")`
`compileOptions(Closure compileOptions)`	Compile options. The argument should be provided as a closure. See the `compileOptions` section for more information.	`compileOptions{ multiCompile.enabled=true outputType="json"}`
`arguments(Map args)`	Additional arguments. These arguments are directly passed to PCT.	`arguments([k1:"v1", k2:'v2'])`

Properties

Property	Required?	Description	Example	Default value
`openedgeVersion`	No (read-only property)	Prints the OpenEdge version. This can be used for logging. This is a read-only property.	None	None

Property	Required?	Description	Example	Default value
`rcodeDir`	No	Sets the directory where the r-code will be generated.	`rcodeDir="path"`	`rcodeDir` set in ABLExtension.
`propath`	No	Sets the PROPATH.	`propath = files('path1', "${dlcHome}/path2")`	`propath` set in ABLExtension
`wrkDir`	No	Sets the working directory.	`wrkDir = 'path'`	`wrkDir` set in ABLExtension
`avmOptions{}`	No	AVM options. See the `avmOptions` section for more information.	None	`avmOptions` set in ABLExtension.
`compileOptions{}`	No	Compile options. See the `compileOptions` section for more information.	None	`compileOptions` set in ABLExtension.
`arguments`	No	Additional arguments. These arguments are directly passed to PCT.	`arguments = [k1:"v1", k2:'v2']`	None

avmOptions{}

Property	Description	Example	Default value
`tmpDir`	The temporary directory for the AVM run time. `-T` is the startup parameter option.	`tmpDir = 'path1'`	None
`tty.enabled`	The flag for using the `_progres` or `prowin` executables. Set to `true` for `_progres`, set to `false` for `prowin` (or `prowin32` for a 32-bit AVM).	`tty { enabled = ‘true’ }`	`true`
`xcodeSessionKey`	The XCODE session key for the secuity policy.	`xcodeSessionKey='myKey'`	None

Property	Description	Example	Default value
parameterFile	The parameter file. -pf is the startup parameter option.	parameterFile='pathToFile'	None
startupParameters	The startup parameters as a string. This will be ignored if a parameterFile is provided	startupParameters='-tok 4000 -s 200'	None
assembliesDir	The assemblies directory. -assemblies is the startup parameter option.	assembliesDir='path1'	None

compileOptions{}

Property	Description	Example	Default value
xrefDir	The path where xref will be generated.	xrefDir='xref2'	'${buildDir}xref'
debugListingDir	The path where the debug listing will be generated.	debugListingDir='debugList2'	'${buildDir}/debugList'
preprocessDir	The path where the preprocess output will be generated.	preprocessDir='preprocess2'	'${buildDir}/preprocess'
multiCompile.enabled	The COMPILER:MULTI-COMPILE attribute	multiCompile.enabled=true	false
strictOptions. requireFullNames	Specifies that the options require full names in the COMPILE statement. Value can be ignore or error.	strictOptions.requireFullNames='error'	ignore
strictOptions. requireFieldQualifiers	Specifies that the options require field qualifiers in the COMPILE statement. Value can be ignore or error.	strictOptions.requireFieldQualifiers='error'	ignore
strictOptions. requireFullKeywords	Specifies that the options require full keywords in the COMPILE statement. Value can be ignore or error.	strictOptions.requireFullKeywords='error'	ignore

Property	Description	Example	Default value
strictOptions. requireReturnValues	Specifies that the options require return values in the COMPILE statement. Value can be ignore or error.	strictOptions.requireReturnValues='error'	ignore
listing.enabled	The LISTING option in the COMPILE statement	listing.enabled=true	false
xcodeKey	The XCODE option in the COMPILE statement. Specify the expression as a string.	xcodeKey='myKey'	None
xref.enabled	The XREF option in COMPILE statement	xref.enabled=true	false
xrefXml.enabled	The XREF-XML option in the COMPILE statement. To enable this option, set the value to true.	xrefXml.enabled=true	false
xrefString.enabled	The STRING-XREF option in the COMPILE statement	xrefString.enabled=true	false
xrefString.append	The APPEND option in STRING-XREF	xrefString.append=true	false
streamIO.enabled	The STREAM-IO option in the COMPILE statement. To enable this option, set the value to true.	streamIO.enabled=true	false
languages.list	This is a comma-separated list of language segments to include in the compiled r-code.	languages.list='lang1,lang2'	None
languages.textSegGrow	The TEXT-SEG-GROW option. Set the growth-factor as an integer. Supported only when a language list is provided.	languages.textSegGrow=10	None
debugList.enabled	The DEBUG-LIST option in the COMPILE statement	debugList.enabled=true	false
preprocess.enabled	The PREPROCESS option in the COMPILE statement	preprocess.enabled=true	false

Property	Description	Example	Default value
v6Frame.enabled	The V6FRAME option in the COMPILE statement	v6Frame.enabled=true	false
v6Frame.useRevVideo	The USE-REVVIDEO option in V6FRAME	v6Frame.useRevVideo=true	false
v6Frame.useUnderline	The USE-UNDERLINE option in V6FRAME	v6Frame.useUnderline=true	false
minSize.enabled	The MIN-SIZE option in the COMPILE statement. To enable this option, set the value to true.	minSize.enabled=true	false
outputType	Use this option to change the output format. Available options are: json	outputType='json'	None

Sample code snippet

The following code snippet is an example using ABLCompile:

```
abl{
    propath("${dlcHome}/tty/")
    avmOptions {}
    compileOptions{}
    rcodeDir='rcode1'
    wrkDir = "myWrkDir"

    dbConnection{
        dbName = 'sports2000'
        connectionParameters='-S 8000'
    }
}

task myDbConTask(type: DBConnection){
    id='id1'
    dbName='sports2020'
    parameterFile='pathToFile'
    aliases = ['alias1', 'alias2', 'aalias3']
}

task myCompileTask(type: ABLCompile){
    println "OE version: ${openedgeVersion}"

    source('src1', 'src2')
    include('**/*.p', '**/*.cls')
    exclude('**/*.txt', '**/*.conf')
    rcodeDir='rcode1'
    propath('path1', "${dlcHome}/path2")
    wrkDir = 'path1'
    avmOptions{
        tty.enabled='true'
        startupParameters='-tok 4000 -s 200'
    }
    compileOptions{
        xrefDir='xref2'
        preprocessDir=""
        preprocess.enabled=""
        strictOptions{
            requireFullNames='warning'
        }
        outputType='json'
    }

    dbConnection{
        dbName = 'sports2020'
        parameterFile='pathToParameterFile'
        connectionParameters='-S 8000'
        port = '8000'
        host = 'localhost'
        username = 'admin'
        password = 'admin2'
        aliases = ['a1', 'a2']
    }
    dbConnectionReferenceId("id0")
    dbConnectionReferenceId("id1", "id2")
}

myCompileTask.dependsOn myDbConTask
```

DBConnection

The DBConnection task defines the OpenEdge database connection, which can be referred to by a compile task.

Properties

Property	Required?	Description	Example	Default value
`dbName`	Yes, if `parameterFile` is not set	The physical database name (`-db` parameter). This can also be a full path.	`dbName = 'sports2020'`	None
`parameterFile`	Yes, if `dbName` is not set	The parameter file	`parameterFile='pathToFile'`	None
`connectionParameters`	Yes, if `dbName` and `parameterFile` are not set	The connection parameters as string. A temporary pf file is created and used as the `parameterFile` property. This property will be ignored if *connection-name*.`parameterFile` is provided	`connectionParameters='-S 8000'`	None
`port`	No	The database port (`-S` parameter)	`port = '8000'`	None
`host`	No	The database host (`-H` parameter)	`host = 'localhost'`	None
`username`	No	The database user name (`-U` parameter)	`username = 'admin'`	None
`password`	No	The database user password (`-P` parameter)	`password = 'admin'`	None
`aliases`	No	The database aliases	`aliases = ['alias1', 'alias2', 'aalias3']`	None
`id`	Yes, if using as a task	The identifier for this task, which can be referred to from a compile task.	`id = 'id1'`	None

Sample code snippet

The following code snippet is an example using DBConnection:

```
task con1(type: DBConnection){
    id = "idCon1"
    dbName = 'sports2020'
    parameterFile='pathToFile'
    aliases = ['a1', 'a2']
}
```

Note:

These properties are passed in the following order:

```
-db <dbName> -pf parameterFile> <other - attributes like -S 8655>
```

The properties that are defined later take priority, if the same parameter is repeated inside the `parameterFile`:

- If `-db` is defined inside the `pf` file and `dbName` is also set, then the property defined in the pf file takes priority.
- If any other parameter, such as `-S 8655`, is defined in the `pf` file and the corresponding attribute, such as *`port`*`='8665'`, is also set, then the attribute, *`port`*, takes priority.

Progress reccommends that you do not define multiple database connection details, because this can lead to unexpected behavior.

ABLRun

Use this task to run ABL procedures.

Methods

Method	Description	Example
`propath(String... propath)`	Sets the AVM's PROPATH. Will be appended to the PROPATHs provided in `ABLExtension`.	`propath('path1', "${dlcHome}/path2")`
`profiler(Closure options)`	Enables profiling. See the `profiler` section for options	`profiler { enabled = true outputDir = "${buildDir}/profiler"}`
`procedure(String procedure)`	Provides the procedure file to be run.	`procedure('src/main.p')`
`environment(String name, String value)`	Adds environment variables to pass to the system command	`environment('k1', 'v1')`
`environment(Map environment)`	Adds environment variables to pass to the system command	`environment([k1:"v2", k2:"v2"])`

Method	Description	Example
`dbConnectionReferenceId(String... refIds)`	Adds database task references. Reference the `id` property of the DBConnection tasks for more information. Note that you must add the database task dependency to this (compile task).	`dbConnectionReferenceId("id1", "id2")`
`dbConnection(Closure connection)`	Adds database configurations. The argument should be provided as a closure with its object following the structure supported by the DBConnection task type. Note that braces can be removed following the Java/Groovy syntax.	`dbConnection{ dbName = 'sports2000' connectionParameters='-S 8000'}`

`avmOptions(Closure avmOptions)`	AVM options. The argument should be provided as a closure.
	See the `avmOptions` section for more information.
		`avmOptions{`
		`tmpDir="path"`
		`tty.enabled=true`
		`}`

`arguments(Map args)`	Additional arguments for ABL procedures. (These arguments are directly passed to PCT).	`arguments([k1:"v1", k2:'v2'])`

Properties

Property	Required?	Description	Example	Default value
`dlcHome`	Yes Should be set as described in the `ABLExtension` section.	OpenEdge DLC path	None	None
`openedgeVersion`	No (read-only property)	Specifies the OpenEdge version. This can be used for logging. This is a read-only property.	None	None

Property	Required?	Description	Example	Default value
`procedure`	Yes	Path of the procedure file that will be run. Can be provided using the method `procedure(String procedure)`.	`procedure = 'src/main.p'`	None
`propath`	No	Sets the AVM's PROPATH. Will be appended to the PROPATHs provided in `ABLExtension`.	`propath = files('path1', "${dlcHome}/path2")`	`propath` set in `ABLExtension`
`wrkDir`	No	Working directory from where the AVM is started	`wrkDir = 'path'`	`wrkDir` set in ABLExtension
`avmOptions`	No	AVM options. Use the method `avmOptions(Closure avmOptions)` to set the properties. The properties defined in this task take priority if the same property is defined in `ABLExtension`	None	`avmOptions` set in `ABLExtension`
`dbConnection`	No	DB connection details. Use methods `dbConnection(Closure connection)` or `dbConnectionRefId(String... refIds)` to add	None	`dbConnections` set in `ABLExtension`

Property	Required?	Description	Example	Default value
`environment`	No	Environment variables to pass to the system command	`environment = [k1:"v2", k2:"v2"]`	None
`arguments`	No	Additional arguments for ABL procedures. (These arguments are directly passed to PCT.)	`arguments = [k1:"v1", k2:'v2']`	None

Profiler

Property	Description	Example	Default value
`enabled`	Enables the profiler	`enabled=true`	`false`
`description`	Description of the profiler session	`description="description"`	None
`outputDir`	Generates a profiler output in this directory, with a unique name. Will be ignored if `outputFile` is provided.	`outputDir = 'path'`	`"${buildDir}/profiler"`
`outputFile`	Profiler output file name	`outputFile='path'`	None
`coverage`	Enables code coverage	`coverage=true`	`false`
`statistics`	Enables statistics	`statistics=true`	`false`
`debugListDir`	Generates debug listing files in this directory	`debugListDir="path"`	None

Sample code snippet

The following code snippet is an example using ABLExtension:

```
task rumMain(type: ABLRun){
    procedure = "src/main.p"
    propath('src')

    wrkDir = "${buildDir}/rcode"
    avmOptions{
        tty.enabled = false
    }

    dbConnection{
        dbName="db/sports2000/sports2000"
        connectionParameters = "-1"
    }

    arguments = [ failOnError : true]
    profiler {
        enabled = true
        coverage = true
        outputDir = "${buildDir}/profiler"
        description = "Coverage for main.p"
    }
}
```

ABLUnit

Use this task to run ABLUnit tests. This task type has all the methods and properties that are in ABLRun (except `procedure`). Additional properties and methods are listed in the following table:

Methods

Method	Description	Example
`exclude(String... excludes)`	Adds an ANT-style exclude pattern	`exclude('**/*.txt', '**/*.conf')`
`include(String... includes)`	Adds an ANT-style include pattern	`include('**/*.p', '**/*.cls')`
`source(String... sources)`	Adds sources to this task, after the include and exclude patterns are applied. The task is skipped if no sources are available.	`source('src1', 'src2')`

Properties

Property	Required?	Description	Example	Default value
`outputDir`	No	Directory where the result file is placed. Don't use this property under Linux, because a bug prevents `results.xml` from being generated	`outputDir="path"`	`"${projectDir}"`

PL

Use this task to create a Procedure Library (PL) file.

Methods

Method	Description	Example
`from(String... from)`	Adds sources and r-codes to this task, after the include and exclude patterns are applied.	`from("${buildDir}/rcode", 'src2')`
`exclude(String... excludes)`	Adds an ANT-style exclude pattern	`exclude('**/*.txt', '**/*.conf')`
`include(String... includes)`	Adds an ANT-style include pattern	`include('**/*.r', '**/*.p')`
`arguments(Map args)`	Additional arguments for libraries. (These arguments are directly passed to PCT.)	`arguments([k1:"v1", k2:'v2'])`

Properties

Property	Required?	Description	Example	Default value
`plFile`	Yes, if `sharedFile` is not provided	The file path where the PL file is created	`plFile="path"`	None
`sharedFile`	Yes, if `plFile` is not provided	Memory mapped library to create the PL file	sharedFile="path"	None

Property	Required?	Description	Example	Default value
`cpInternal`	No	Internal code page (`-cpinternal` parameter)	`cpInternal="value"`	None
`arguments`	No	Additional arguments for libraries. (These arguments are directly passed to PCT.)	`arguments = [k1:"v1", k2:'v2']`	None

Sample code snippet

The following code snippet is an example using PL:

```
task createPL2(type: PL){
    plFile = "${buildDir}/dist/test.pl"
    from("${buildDir}/rcode")
    include('**/*.r')
    arguments = [encoding  : 'undefined' ]
}
```

BackupDB

Use this task to create a backup of an OpenEdge database.

Properties

Property	Required?	Description	Example	Default value
`source`	Yes	Path to the database where the `.db` file is located. Can be relative or absolute. May or may not include the `.db` file extension.	`source="path"`	None
`target`	Yes	Path of the backup file for the database. Can be absolute or relative.	`target="path"`	None
`overwrite`	No	Whether to replace if the backup file exists	`overwrite=true`	`false`

Property	Required?	Description	Example	Default value
online	No	Indicates the database must be online or not when doing a backup. If not set, the proutil -C holder is used to detect if the database is up. Leave this out if you want the task to auto detect if this option should be used.	online=true	Auto
incremental	No	Performs an incremental backup.	incremental=true	false

Sample code snippet

The following code snippet is an example using BackupDB:

```
task createDBBackup(type: BackupDB){
    source = "db/sports2020/sports2020.db"
    target = "${buildDir}/dist/sports2020.bck"
    overwrite = true
}
```

CreateDB

Use this task to create an OpenEdge database.

Methods

Method	Description	Example
arguments(Map args)	Additional arguments. These arguments are directly passed to PCT.	arguments([k1:"v1", k2:'v2'])

Properties

Property	Required?	Description	Example	Default value
dbName	Yes	Database name	dbName="dbName"	None
outputDir	No	The directory where the database will be created	outputDir="path"	"${projectDir}"

Property	Required?	Description	Example	Default value
`sourceDb`	No	Copy the specified database to the target database.	`sourceDb="path"`	If attribute is not provided, the empty DB is used
`schemaFile`	No	Initial dump files to load into database. Separate dump filenames with commas. Files are resolved first as an absolute path, then relative to base directory. Wildcards are not expanded.	`schemaFile="path"`	None
`structFile`	No	Structure description file	`structFile="path"`	None
`blockSize`	No	Block size in kilobytes (1, 2, 4, or 8). Cannot be used with `sourceDb` attribute.	`blockSize=2`	8
`tmpDir`	No	The `-T` parameter when loading a schema	`tmpDir="path"`	None
`cpInternal`	No	The `-cpinternal` parameter when loading a schema	`cpInternal="value"`	None
`newInstance`	No	Appends `-newInstance` in the PROCOPY command line.	`newInstance=true`	`false`
`largeFiles`	No	Enable large files for this database.	`largeFiles=true`	`false`
`arguments`	No	Additional arguments for databases. (These arguments are directly passed to PCT.)	`arguments = [k1:"v1", k2:'v2']`	None

Sample code snippet

The following code snippet is an example using CreateDB:

```
task createSports2000(type: CreateDB){
    dbName = 'sports2000'
    sourceDb = "${dlcHome}/sports2000"
    outputDir = "db/sports2000"
}
```

LoadDBSchema

Use this task to load an OpenEdge database schema file to a database.

Methods

Method	Description	Example
`exclude(String... excludes)`	Adds an ANT-style exclude pattern.	`exclude('**/*.txt', '**/*.conf')`
`include(String... includes)`	Adds an ANT-style include pattern.	`include('**/*.df')`
`source(String... sources)`	The schema (`.df`) files to be loaded. Adds sources to this task after the include and exclude patterns are applied. The task is skipped if no sources are present.	`source('myDb.df', 'src/schema')`
`dbConnection(Closure connection)`	Adds database configurations. The argument should be provided as a closure with its object following the structure supported by DBConnection task type. Note that braces can be removed following Java/Groovy syntax.	`dbConnection{ dbName = 'sports2000' connectionParameters='-S 8000'}`
`dbConnectionReferenceId(String... refIds)`	Adds database tasks' references. Reference the `id` property of the DBConnection tasks for more information. Note that you mustadd the Database task dependency to this (compile task).	`dbConnectionReferenceId("id1", "id2")`
`arguments(Map args)`	Additional arguments for dtaabases. (These arguments are directly passed to PCT.)	`arguments([k1:"v1", k2:'v2'])`

Properties

Property	Required?	Description	Example	Default value
onlineChanges	No	Relaxes requirements in order to apply online changes (use SESSION:SCHEMA-CHANGE = 'NEW OBJECTS').	onlineChanges=true	false

Sample code snippet

The following code snippet is an example using LoadDBSchema:

```
task loadSports2000Schema(type: LoadDBSchema){
    source("myDb.df")
    onlineChanges = true
    dbConnection{
        dbName="db/sports2020/sports2020"
        connectionParameters = "-1"
    }
}
```

Sample custom task

The following sample code demonstrates how you can use the ABLCompile and DBConnection task types in a custom task:

```
import com.progress.gradle.abl.tasks.*
plugins {
  id "progress.openedge.abl-base" version "1.0.0"
}

// This section defines the global configuration available to all task types.

abl{
    propath("${dlcHome}/tty/")
    avmOptions {}
    compileOptions{}
    rcodeDir='rcode1'
    wrkDir = "myWrkDir"

    dbConnection{
        dbName = 'sports2000'
        connectionParameters='-S 8000'
    }
}

// This section defines the DBConnection task.

task myDbConTask(type: DBConnection){
    id='id1'
    dbName='sports2020'
    parameterFile='pathToFile'
    aliases = ['alias1', 'alias2', 'aalias3']
}

// This section defines the ABLCompile task.

task myCompileTask(type: ABLCompile){
    println "OE version: ${openedgeVersion}"

    source('src1', 'src2')
    include('**/*.p', '**/*.cls')
    exclude('**/*.txt', '**/*.conf')
    rcodeDir='rcode1'
    propath('path1', "${dlcHome}/path2")
    wrkDir = 'path1'
    avmOptions{
        tty.enabled='true'
        startupParameters='-tok 4000 -s 200'
    }
    compileOptions{
        xrefDir='xref2'
        preprocessDir=""
        preprocess.enabled=""
        strictOptions{
            requireFullNames='warning'
        }
        outputType='json'
    }

    dbConnection{
        dbName = 'sports2020'
        parameterFile='pathToParameterFile'
        connectionParameters='-S 8000'
        port = '8000'
        host = 'localhost'
        username = 'admin'
        password = 'admin2'
        aliases = ['alias1', 'alias2', 'aalias3']
    }
    dbConnectionReferenceId("id0")
    dbConnectionReferenceId("id1", "id2")
}

myCompileTask.dependsOn myDbConTask
```

Migrate from Apache Ant

Many ABL applications make use of Apache Ant and PCT. Apache Ant is a build automation tool that uses XML and Java, and it was designed to make build processes simpler and more portable. Ant uses targets and tasks to compile, test, and run applications. It is primarily used for Java applications, but can be used for other types of applications as well. PCT is an open source set of Apache Ant tasks and plugins created by Riverside Software that automate work in OpenEdge. PCT is a large ecosystem of plugins that provides common build steps for automation and CI/CD.

Why migrate from Ant to Gradle?

Migrating from Ant to Gradle has the following benefits:

- Gradle uses a domain-specific language, which allows users to add logic to the build pipeline.
- Gradle has reusable tasks and can reuse other parts of a build pipeline, which makes the build script less verbose, more readable, and easier to maintain.
- Gradle provides smarter dependency management than Ant.
- Gradle make builds modular, which allows larger projects to be built from interdependent parts. You can even build multiple projects from the same pool of parts.

Use ant.importBuild

The Gradle task `ant.ImportBuild` is a straightforward and quick way to migrate your build processes from PCT or Ant to Gradle. This task converts Ant targets in your Ant scripts into Gradle tasks. To use `ant.ImportBuild`:

1. Add the following to your `build.gradle` file in your project:

   ```
   ant.ImportBuild 'build.xml'
   ```

   This assumes `build.xml` contains your previous Ant tasks.

2. After saving your `build.gradle` file, run your previous Ant tasks using the Gradle wrapper:

   ```
   gradlew ant target
   ```

Using `ant.ImportBuild` works quite well as an initial phase of migrating to Gradle, but it is not a full migration because you still need to maintain your existing Ant scripts.