



## **Whats New in OpenEdge DevOps Framework**



# Table of Contents

Copyright.....	5
Learn about OpenEdge DevOps Framework.....	7
What's New in OpenEdge DevOps Framework 2.3.....	9
What's New in OpenEdge DevOps Framework 2.2.....	11
What's New in OpenEdge DevOps Framework 2.1.....	13



# Copyright

---

Visit the following page online to see Progress Software Corporation's current Product Documentation Copyright Notice/Trademark Legend: <https://www.progress.com/legal/documentation-copyright>.

**December 2023**

**Product version:** Progress OpenEdge DevOps Framework 2.3



## Learn about OpenEdge DevOps Framework

---

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.



## What's New in OpenEdge DevOps Framework 2.3

---

### Support for Gradle 8

The OpenEdge DevOps Framework now supports Gradle 8. For more information, see [What's new in Gradle 8.0](#), [Upgrading your build from Gradle 8.x to the latest](#), and "Prerequisites" in the *OpenEdge DevOps Framework Guide*.

---

**Note:** If you are using the legacy `apply()` method to use the OpenEdge DevOps Framework 2.3 plugin then note that there is a change in the classpath of the plugin. For more information, see [progress.openedge.abl.base](#) and [progress.openedge.abl](#).

---



## What's New in OpenEdge DevOps Framework 2.2

---

### **Generate Deployable Packages for PAS for OpenEdge Application**

OpenEdge DevOps Engineers and OpenEdge Developers can now create different deployable packages like \*.oear, \*.war, \*.paar, and \*.oeds files.

This feature provides OEDF the capability to pack the build output artifacts as a standard deployable package, which the DevOps Engineer or OpenEdge Developer can directly deploy to the Progress Application Server (PAS) for OpenEdge instance.

For more information about creating deployable packages for PAS for OpenEdge application, see "Build packages for PAS for OpenEdge application" in the *OpenEdge DevOps Framework Guide*.



## What's New in OpenEdge DevOps Framework 2.1

---

Progress OpenEdge DevOps Framework 2.1 includes the following new feature:

- **Dependency management and resolution in OEDF**—OpenEdge DevOps engineers can now efficiently manage the internal and external software dependencies of their ABL application using the automatic dependency resolution and management capability in the OpenEdge DevOps Framework gradle plugins.

With this capability, you no longer need to use the traditional way of configuring the application dependencies at different places for builds or tests. Instead, you can now specify various dependencies required by your application, their path (public repository, private repository, local or network files system, and so on), and the scope of the dependency (compile or test) at one place. When a build or test script is executed, the dependencies are automatically fetched from the specified repositories and used for the associated compile or test activity.

