Progress®



OpenEdge Continuous Integration and Delivery

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: 2021/06/30

Preface

Purpose

This manual describes how to prepare and use OpenEdge applications with continuous integration and continuous delivery applications, features, and software. It describes the options within OpenEdge, third party options, and recommended best practices.

Audience

This manual is intended for OpenEdge developers and OpenEdge system administrators who are familiar with creating and maintaining OpenEdge applications. Familiarity with CI/CD is beneficial, but not required to understand the information in this manual. Additional learning sources are referenced throughout this manual.

Documentation conventions

See Documentation Conventions for an explanation of the terminology, format, and typographical conventions used throughout the OpenEdge content library.

Learn about CI/CD with OpenEdge

For details, see the following topics:

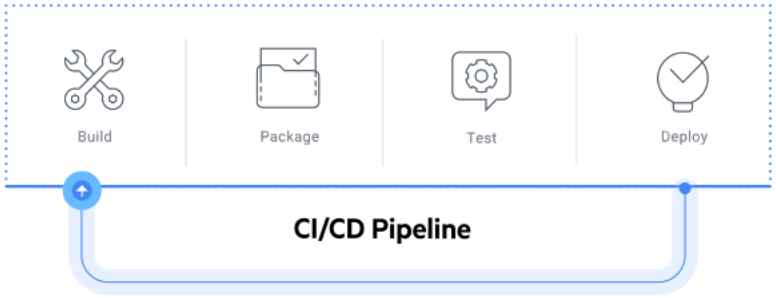
- Learn about CI/CD

Learn about CI/CD

CI/CD stands for continuous integration and continuous delivery or continuous deployment. A CI/CD pipeline is an automated process for software delivery that combines the development, testing, and delivery processes into one, seamless system. CI/CD improves product quality, release frequency, and release predictability. CI is the method that merges code changes back to the main branch as quickly as possible. CD supports the release of new changes to your customers quickly. This means automating your testing and release process. CD is also the method that automatically releases every change that passes all stages of your production pipeline to your customers. Therefore, continuous integration is the first step for both continuous delivery and continuous deployment.

In an ideal CI/CD pipeline, an automated process can build, package, and test code every time a developer checks in code changes, which allows for flexibility and agility in software development. The goal of implementing a CI/CD pipeline is to build and test software frequently and to continuously deploy to an environment. When you implement a CI/CD pipeline, you benefit by reducing risk during deployment, having a system of continuous feedback, improving code coverage, increasing quality, and improving productivity.

There are four core phases of a CI/CD pipeline:



1. The build phase, where the source code is built, statically analyzed, and unit tested
2. The package phase, where the code is packaged into a working component and pushed to an artifact repository
3. The test phase, where the artifact is tested to ensure that it functions correctly, including functional and integration tests
4. The deploy phase, where the artifact is made available for delivery

After deployment, a CI/CD pipeline circles back to the build phase as you continue to work on the code. The cycle of a CI/CD pipeline can continue perpetually, continuously integrating new code, and delivering and deploying updated software.

OpenEdge is designed to fit into many automation solutions. OpenEdge 12 includes a set of tools and solutions that help you move the development of your ABL applications into a CI/CD pipeline. This guide describes the recommended CI/CD options for OpenEdge, to help you automate your development and deployment across all stages of a CI/CD pipeline.

The OpenEdge features that facilitate an automated pipeline are:

- 'ABLUnit' in Progress Developer Studio for OpenEdge Online Help
- Build with Apache Ant on page 13
- Build with Gradle on page 14
- 'ABL procedure libraries' in *Manage ABL Applications*
- Create containers on page 17
- 'OpenEdge Application Archive' in *Manage Progress Application Server (PAS) for OpenEdge*

Modernize your OpenEdge application

Working with your codebase to facilitate automation and CI/CD is a key component of creating a modern devops process for software applications. Consider updating several components of your application, to smoothly implement your automation strategy. For more information about best practices and recommendations for how to modernize your OpenEdge application, see *Modernize Progress OpenEdge Applications*.

Feedback loops

Feedback loops are an integral part of automation and CI/CD. Your CI/CD pipeline does not end at deployment; it iterates through the pipeline as long as you are developing software. Engaging in feedback loops provides you with many opportunities to adjust your software and code to meet evolving customer needs. This is why automation is key when you adopt a CI/CD pipeline. As more phases of the pipeline are automated, the faster you can complete an entire cycle. Eventually, you may have a CI/CD pipeline process that completes with only one input from building through deployment.

Quality gates

Quality gates are thresholds that software needs to reach in order to move along a CI/CD pipeline. One example of a quality gate is the percentage of tests that must pass in order to move to the next stage. This quality gate could be configured to automatically fail an application that has less than a 90% pass rate for all tests. Failing the pass rate stops the application during the test phase, and causes the application to be sent back to the build phase. Quality gates and testing phases can be set up at all stages of a CI/CD pipeline.

Infrastructure as Code

Infrastructure as Code is a conceptual framework for managing the data infrastructure as though it were a codebase. Infrastructure as Code is a key part of CI/CD because it allows the infrastructure that surrounds software development to be automated, enabling the entire CI/CD process. Automating the infrastructure increases the speed and agility of development, increases efficiency with defined processes, and mitigates human error, thus reducing risk.

II

Build

For details, see the following topics:

- Coding practices
- Build with Apache Ant
- Build with Gradle
- Third-party testing tools

Coding practices

Establishing standard coding practices is the first step to automating your OpenEdge development

Source control

Source control is an important part of a CI/CD pipeline, and it enables collaboration among software developers. Source control ensures that there is a single source of truth when it comes to the code. With multiple developers working on the same code, merging and managing their disparate changes can be confusing, and can cause errors and regressions. Using a development platform with a source control system mitigates the risk of developers overwriting others' changes, preventing many potential disasters.

There are a number of options for source control, including Roundtable TSMS, GitHub, GitLab, and Apache Subversion.

ABLUnit test framework

The ABLUnit test framework is used for unit testing ABL code. The framework is provided with Progress Developer Studio for OpenEdge, but it can also be used with projects outside of Progress Developer Studio.

For more information, see 'Learn About ABLUnit Test Framework' in *Progress Developer Studio for OpenEdge Online Help*.

Architectural components

The following are the core set of logical components that make up an OpenEdge application. Additional logical components may also be included in your application:

- Service interfaces
- Business logic
- Data stores
- Common libraries, including `$DLC`

A service requires two components: an API and a service interface. In an OpenEdge application, an ABL WebApp provides the API, and the ABL service provides service interfaces. The service interface communicates and interprets requests from all aspects of the application and transforms the incoming data. The service interface is a translation layer that communicates requests to the appropriate business services through business logic. Service interfaces can also provide authentication, authorization, and error handling.

The business logic, data stores, and common libraries are the core pieces of the application, and are what is traditionally thought of as an application. Business logic is made up of things like master data maintenance, order entry, or tax calculations. Data is stored in one or more OpenEdge databases. Common or shared libraries contain generic code that is needed for an application. Configuration files for deployment, authorization, and authentication can also be considered core pieces of an appliaction.

Unit testing

Unit testing is the process of testing individual components of a software application to ensure that they work in isolation. This is the simplest stage of testing because it does not test an application as a whole; it tests individual sections of code. Unit testing is helpful when the code is being developed because it can catch issues very early. Unit testing is especially beneficial when it is automated because you can quickly determine if the code you just wrote is functional. If there are problems with the code, then you can make changes quickly.

API testing

API testing is a subsection of integration testing that focuses on ensuring that the APIs for your software are functioning correctly. APIs are tested individually, similar to unit testing, and in transactions to ensure they function properly in isolation and together. Notable API testing tools include Fiddler and Postman.

R-code

R-code is binary code that is created when OpenEdge compiles ABL code. R-code is a secure code format generated as r-code files (`.r` files) from ABL source code, whether encrypted or in plain text.

For more information, see 'R-code Features and Functions' and 'R-code structure' in *Manage ABL Applications*.

Procedure libraries

A procedure library (`.pl` file) is a collection of files and images that is used to package an ABL application. Procedure libraries are used to both efficiently store and efficiently execute r-code files and ABL code. You can use procedure libraries with many file types, such as `.r`, `.wrx`, `.w`, and `.i` files.

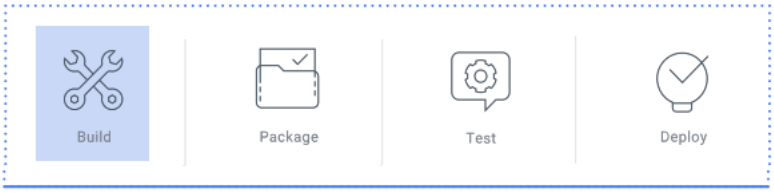
For more information, see 'Manage Procedure Libraries' and 'Library overview' in *Manage ABL Applications*.

Encrypt source code with XCODE

The XCODE utility is provided with ABL, and enables you to encrypt your ABL source code. Encrypting your source code provides application security because users cannot read your application source code or make changes to it. XCODE uses less storage than r-code and is therefore much more portable, with the added benefit of still allowing compilation into r-code files.

For more information, see 'XCODE utility' and 'Prepare to use XCODE' in *Manage ABL Applications*.

Build with Apache Ant



CI/CD Pipeline

Apache Ant is a build automation tool that uses XML and Java, and it is designed to make build processes simpler and more portable. Ant uses targets and tasks to compile, test, and run applications. Ant is primarily used for Java applications, but it can be used for other types of applications.

Ant tasks for OpenEdge

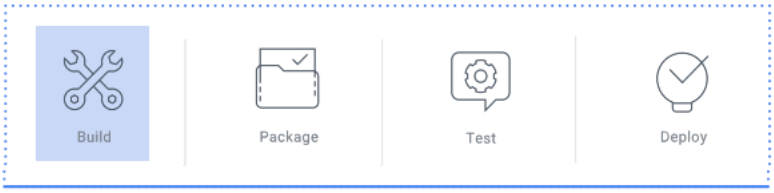
The following procedures can be performed using Progress-provided Ant tasks for OpenEdge:

- 'Generate ABLDoc using Ant' in *Progress Developer Studio for OpenEdge Online Help*
- 'Package an ABL WebApp Ant project' in *Progress Developer Studio for OpenEdge Online Help*
- 'Package REST services' in *Progress Developer Studio for OpenEdge Online Help*
- 'Generate a Data Object Service Catalog file' in *Progress Developer Studio for OpenEdge Online Help*
- 'Package a Web UI project' in *Progress Developer Studio for OpenEdge Online Help*
- 'Unit test with ABLUnit' in *Progress Developer Studio for OpenEdge Online Help*

PCT

Progress Compilation Tools (PCT) is an open-source set of Apache Ant tasks and plugins created by Riverside Software that automate build tasks in OpenEdge. PCT is a large ecosystem of plugins that provides common build steps for automation and CI/CD. Progress recommends PCT as the method for automating OpenEdge through Ant, and many Ant processes with OpenEdge use it as a standard foundation. You can use Ant with OpenEdge without using PCT, but that requires that you define a custom Ant implementation for your CI/CD pipeline.

Build with Gradle



CI/CD Pipeline

Gradle is an open-source build automation tool that is an evolution of the concepts used in Apache Ant. Gradle intelligently determines which parts of a codebase have not changed, and only builds and runs the parts that have changed. This process allows builds to be smarter, faster, and more efficient because it eliminates redundancies in the build phase. For more information about Gradle, see the Gradle documentation.

OpenEdge DevOps Framework

The OpenEdge DevOps Framework is a Gradle plugin for OpenEdge applications that supports building projects across platforms, and enables smooth transitions between the phases of a CI/CD pipeline. For ABL applications, the OpenEdge DevOps Framework helps implement an efficient CI/CD 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 both users who are new to the CI/CD process and advanced DevOps engineers with a complex CI/CD process and additional flexibility needs.

For more information, see Unified Build and Packaging Framework in *Progress Developer Studio for OpenEdge Online Help*.

For more information, see 'Unified Build and Packaging Framework' in *Progress Developer Studio for OpenEdge Online Help*.

Third-party testing tools

There are many third-party testing options available for your OpenEdge applications. Consider using one of the following options to enhance the test phase of your CI/CD pipeline.

SonarQube

SonarQube is a tool that scans and reviews your codebase and reports potential issues. There are several OpenEdge plugins available for SonarQube, including some created by Riverside Software, the creators of PCT. For more information about SonarQube, see the SonarQube website.

Veracode

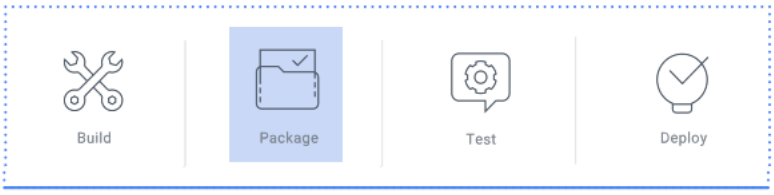
Veracode provides a cloud-based security platform that helps prevent vulnerabilities in your software. Veracode analyzes your codebase on several axes to provide a comprehensive security overview. You must customize Veracode to use it with ABL, but it is an option for .NET or Java Open Clients. For more information about Veracode, see the Veracode website.

Package

For details, see the following topics:

- Package ABL applications
- Create an OpenEdge Application Archive
- Create containers

Package ABL applications



CI/CD Pipeline

Packaging uses all output from the build phase to create consumable artifacts that can be tested and deployed to the customer. These artifacts include r-code, procedure libraries, configuration files, service definitions, online help, and other options.

Package overview

The output artifacts, code, and services generated from the build phase are combined to form a component. Components are the smallest logical units you can use to define the parts of your ABL business application. Components are be packaged together to build your business application.

As applications are packaged for deployment, you often combine components with configuration and settings. Each package should be independently developed, built, and versioned. Packages should not be tied to your OpenEdge version because changes made in OpenEdge versions could disrupt your packages. It is best for your packages to contain data and code that is independent of your OpenEdge version. Packages should be public and visible so they can be accessed easily within a CI/CD pipeline.

The following development artifacts are commonly packaged together to build OpenEdge application deployment artifacts:

- ABL code and data
- Static resources such as images and web pages
- Configuration files
- Scripts and procedures
- Environment
- Metadata

Application configurations

The simplest, runnable Progress Application Server for OpenEdge configuration contains one instance, one application, one WebApp, and one or more services. This basic configuration is what you get by default when creating an instance with `tcman create`. However, an instance can support many applications, depending on the available CPU and memory. When using one application per instance, you have advantages in scalability, flexibility, and simplicity. When using many applications per instance, you can have a family of applications that performs different functions within your instance. Having multiple applications per instance is also useful for developer environments because it allows you to do smaller deployments and share resources between applications. In addition to having multiple applications within a single instance, you could have multiple WebApps within a single application. This configuration allows you to use multiple services with each other in a more structured way. WebApps can contain many ABL services.

The best application configuration can be determined through load testing and identifying the right balance of applications, WebApps, and services. Another consideration for multi-tenant hosted systems is being sensitive to Service Level Agreements that can differ depending on the size of your operation. You might have smaller customers sharing an instance and provide large customers their own instances. Load balancing is another consideration when defining your instances and applications.

Create an OpenEdge Application Archive

An 'OpenEdge Application Archive' in *Manage Progress Application Server (PAS) for OpenEdge* (OEAR file) can be created for a Progress Application Server (PAS) for OpenEdge application. An OpenEdge Application Archive is a structured archive file that is designed to package ABL applications. The structure of an OEAR file is designed to make packaging ABL applications a simple process, but the structure can be customized as needed. An OEAR file contains web application resources and can be deployed to PAS for OpenEdge in a single operation. Packaging an OpenEdge application into an archive file is the simplest way to prepare your application for deployment. You can customize an OpenEdge Application Archive using the Ant build scripts that come with OpenEdge. Using the provided scripts, you can automate the packaging process for your OpenEdge application.

There are two main methods of converting your OpenEdge application into an OEAR file:

- 'Create an OpenEdge Application Archive using tcman export' in *Manage Progress Application Server (PAS) for OpenEdge*
- 'Create an OpenEdge Application Archive using an Ant Build' in *Manage Progress Application Server (PAS) for OpenEdge*

Create containers

Containers enable developers to deploy and upgrade applications in a unified and simple process. Containers can automatically be created and destroyed to meet demand, providing scalability and flexibility. Containers also make it simple to use applications across many machines and systems. For more conceptual information about containers, see What is a Container?

PAS for OpenEdge container

Progress provides a PAS for OpenEdge container. Running PAS for OpenEdge in a container is beneficial because both the application server instance and the applications deployed to the instance can be updated without affecting other running OpenEdge components.

For more information, see the 'PAS for OpenEdge container guide' in *Run PAS for OpenEdge in a Docker Container*.

OpenEdge database container

Progress provides an OpenEdge database container that is intended for the development and testing of ABL applications. Having a database in a container is not recommended for commercial use, but it makes setting up new testing environments for developers much simpler.

For more information, see 'Why use an OpenEdge database in a Docker container?' in *Learn About OpenEdge Database Docker Containers*

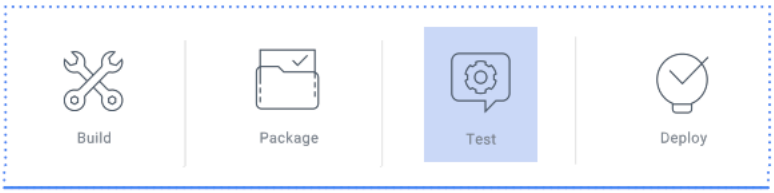
IV

Test

For details, see the following topics:

- Learn about testing concepts

Learn about testing concepts



CI/CD Pipeline

The goal of the test phase is to evaluate your code and ensure that your application functions as intended. This testing should be automated using one of many approaches and types of tests. Progress recommends the following types of tests and testing concepts.

Automated testing

Automation is an important part of a CI/CD pipeline, and automated tests greatly improve the efficiency of a CI/CD pipeline. Automated tests are faster and more consistent than manual tests because they mitigate human error. Automated tests free time for developers who could be coding and improving software instead of manually running tests. Automating tests also provides some flexibility with when and how you test. With automation, you could run tests after each phase of the CI/CD pipeline. You could test after building, after packaging, and finally after deployment.

Integration testing

Integration testing is a process that tests various components and sections of a software application to ensure that they work together. Compared to unit testing, which tests how individual components work in isolation, integration testing tests whether the software works together as a whole. Integration testing is crucial for ensuring that your software works cohesively and for preventing catastrophic failures during later stages of production.

Load testing

Load testing is a process that evaluates how much load a software system can handle. The load can be measured by users, clients, instances, and many other metrics. Load testing is important to gauge how many concurrent instances of your application you can maintain and for how long. Notable third-party options for load testing are SoapUI and Apache JMeter.

ABLUnit test framework

ABLUnit is a unit testing framework for the ABL programs. ABLUnit helps you write and run repeatable unit test cases for automated testing. ABLUnit can be used with Progress Developer Studio and the OpenEdge DevOps Framework.

For more information, see Learn About ABLUnit Test Framework in *Progress Developer Studio for OpenEdge Online Help*.

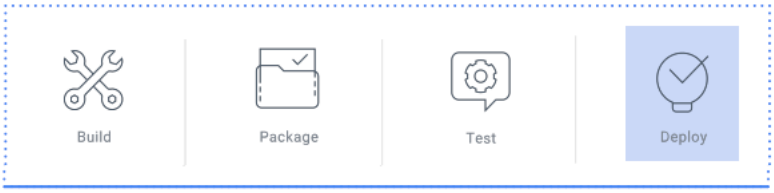
V

Deploy

For details, see the following topics:

- Learn about deployment strategies
- Deploy using TCMAN import

Learn about deployment strategies



CI/CD Pipeline

Deployment is the final phase of a CI/CD pipeline, and is when the software you create is released. Consider the following deployment concepts and strategies for your CI/CD pipeline implementation.

Automatic deployment

In an ideal CI/CD pipeline, after your software is through the test phase, you should be able to automatically deploy and deliver without any input. Getting to that level requires a robust and mature CI/CD pipeline implementation. Automatic delivery benefits both the developers and the consumers. For developers, there is less room for human error and less time performing manual operations. For consumers of your software, they get access to the latest build changes quickly. Many CI/CD pipelines go through the deployment phase without automatically pushing out the software to customers. Your business application is ready for release at any time.

Application deployment levels

Consider the following levels of an OpenEdge application when determining your deployment strategy. You can deploy at all levels of an application, but depending on your business needs, one may provide advantages over another:

1. Install—The base level of your application. Changes at this level affect all other levels. This is the level where `$DLC` and `$CATALINA_HOME` are located.
2. Progress Application Server for OpenEdge instance—The instance runs one or more business applications and potentially Tomcat web applications. The instance level is where load balancing and failover occur.
3. ABL application—A business application that contains one or more WebApps.
4. ABL WebApp—A secure set of services that provide access to the ABL application. A WebApp provides name and authentication services.
5. ABL service—A service interface to the application business logic. Can provide authorization in the WebApp and in ABL code.

Creation and deployment methods for OpenEdge applications

The following methods are provided for creating and deploying applications with OpenEdge.

Table 1: Application methods

Application component	Associated file type	Creation method	Deployment method
Instance	`.zip`	Numerous methods	`tcman register`
Application	`.oear`	`tcman export` Ant scripts	`tcman import`
WebApp	`.war`	Progress Developer Studio Export Ant scripts	`tcman deploy`
Service	Not applicable	Progress Developer Studio Export	`deploySvc`

Deploy using TCMAN import

After you automate the package step of your OpenEdge application process, you can use the provided OpenEdge Application Archive scripts to automate deployment. The TCMAN command-line utility is used to manage Progress Application Server (PAS) for OpenEdge. You can deploy your application using the TCMAN `import` command. TCMAN `import` brings an application packaged as an `.oear` file into a PAS for OpenEdge instance.

For more information, see 'Deploy an OpenEdge Application Archive using tcman import' in *Manage Progress Application Server (PAS) for OpenEdge* and 'Import an ABL application (import)' in *PAS for OpenEdge Administration Tools Reference*.

You can perform many other tasks using TCMAN operations. For more information, see 'TCMAN Reference' in *PAS for OpenEdge Administration Tools Reference*.

Customize with Ant

You can customize your TCMAN `import` deployment using Ant scripts and tasks to create and modify a `build.xml` file. The default `build.xml` file can be used as a template, or you can create your own.

For more information, see 'Tailor an ABLApp installation using Ant' in *Manage Progress Application Server (PAS) for OpenEdge*.

VI

Learn more

For details, see the following topics:

- Related solutions

Related solutions

Automation is just one of many steps in creating a modern OpenEdge application. For more options, see our other solutions guides:

- *Modernize Progress OpenEdge Applications*
- *Achieve Continuous Operations with OpenEdge*
- *Optimize OpenEdge Performance*
- *Authenticate Users Defined in Microsoft Active Directory*