Progress®

Learn about Progress Application Server for OpenEdge

21 June 2023

Copyright

Table of Contents

1

What is PAS for OpenEdge?

Progress Application Server for OpenEdge (PAS for OpenEdge) is an Apache Tomcat web server that was extended to support OpenEdge and ABL. PAS for OpenEdge has built-in routing and adapter functionality that enables you to access your business logic on a variety of clients including ABL, browser-based clients, and mobile clients. Progress development applies Apache Tomcat updates that ensure compliance with industry standards, like Spring Security, while supporting all client types in a single server using easy-to-configure properties files. Because PAS for OpenEdge is an application server, it makes implementing and deploying applications efficient and simplifies application management.

To use PAS for OpenEdge, you:

- Create and configure instances.
- Deploy ABL or WebSpeed application code to an instance.
- Deploy REST and SOAP services to an instance.
- Create and deploy web handlers for application modernization.
- Start, stop, and delete instances.
- Deploy management applications such as Tomcat manager and OpenEdge manager to an instance.
- Monitor instances.
- Implement load balancing.
- Implement security such as HTTPS, user authentication and authorization.

Application development, testing, debugging, and deployment is supported by Progress Developer Studio for OpenEdge, which includes a project type and a development server for PAS for OpenEdge. Progress Developer Studio is the recommended development tool for web applications deployed on PAS for OpenEdge.

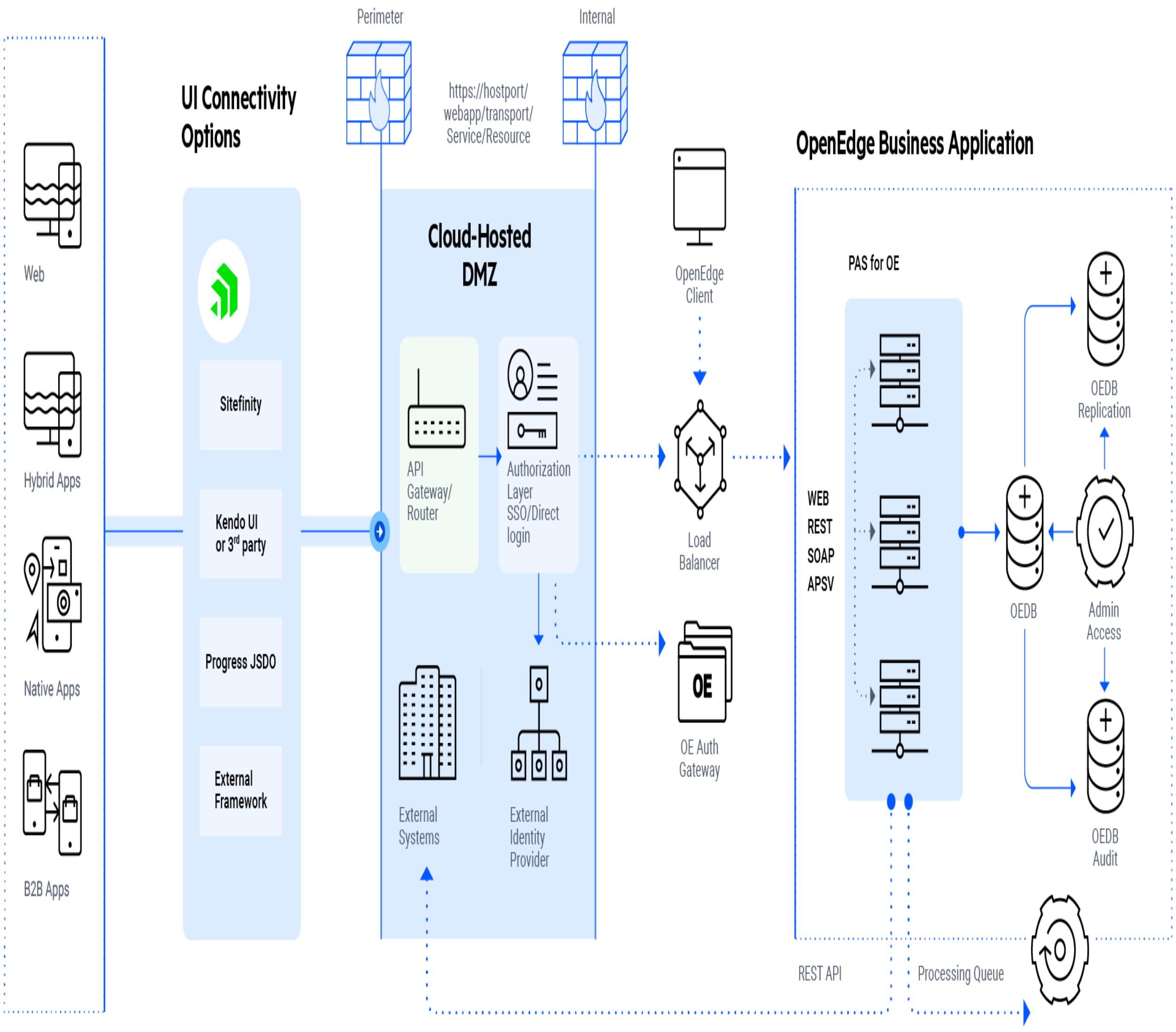
For more information, see Learn About PAS for OpenEdge in *Progress Developer Studio for OpenEdge Online Help*.

Related Links

- PAS for OpenEdge architecture
- Instances
- ABL applications
- The ROOT application
- Web applications
- Transports and services
- Manager applications
- Catalina environment variables
- PAS for OpenEdge directories
- HTTP sessions
- Property files
- Log files
- Use PAS for OpenEdge with Docker containers
- PAS for OpenEdge Lite
- Main steps: Install and use PAS for OpenEdge

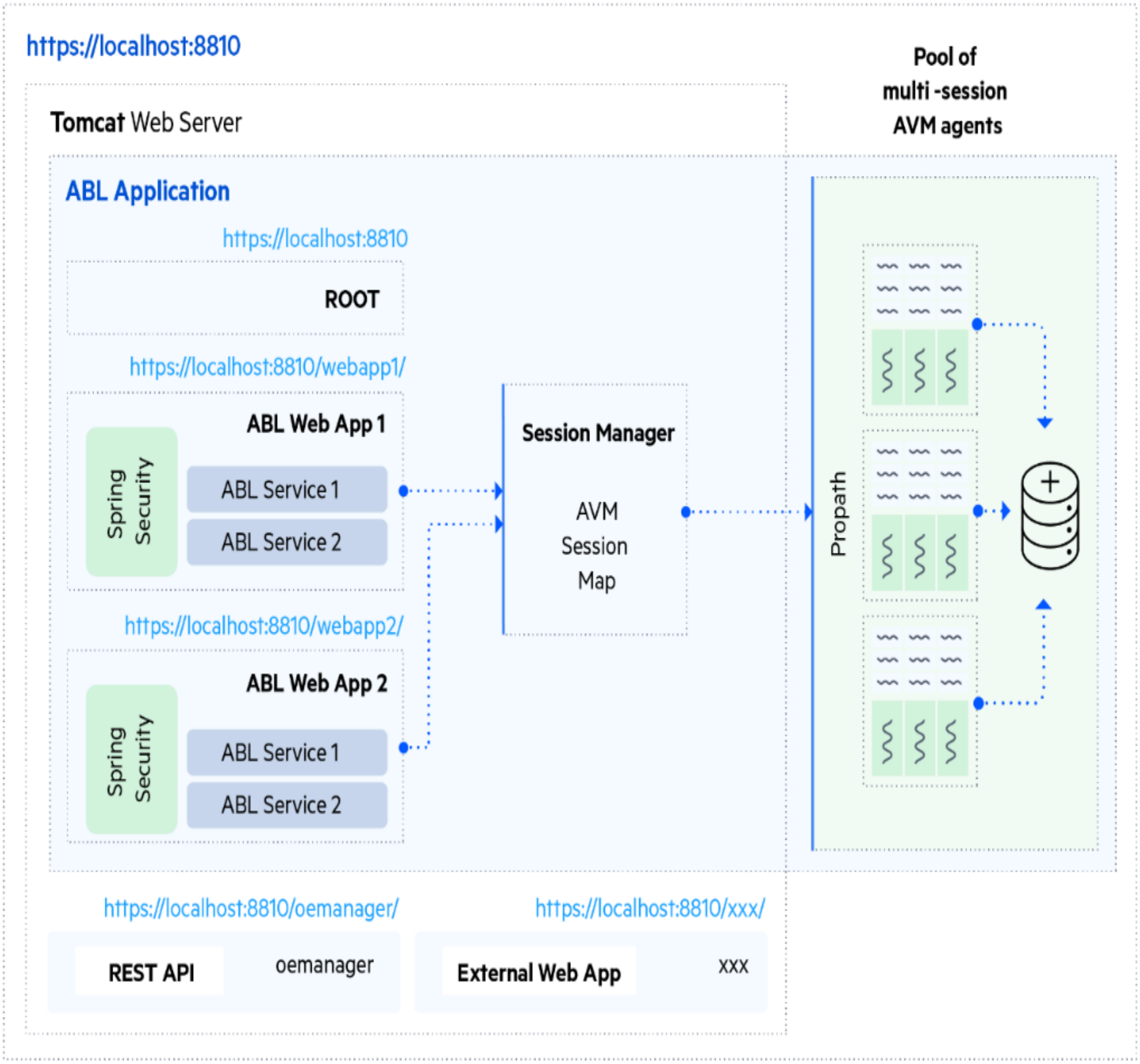
PAS for OpenEdge architecture

PAS for OpenEdge is a component of the OpenEdge environment. The following figure shows PAS for OpenEdge and the related OpenEdge components:



PAS for OpenEdge is the application server component of an OpenEdge business application. PAS for OpenEdge interacts with the OpenEdge database and serves client requests using transports.

Each PAS for OpenEdge instance has the following components, as shown:



The components in the diagram are:

- The Tomcat Web Server (PAS for OpenEdge Instance). Tomcat is used as both a web server to deliver static web content and as an application server for ABL applications. Tomcat accepts incoming HTTP/S requests from clients such as ABL, browser clients and mobile devices and routes those to the web applications deployed to the Tomcat web server. All clients (including ABL and OpenClient) communicate with PAS for OpenEdge using HTTP. You can find details on the Tomcat environment at PAS for OpenEdge Instances.
- An ABL application is a grouping of ABL web applications and business logic that operate within a unique PROPATH, set of database connections, application configuration such as locale, and security configuration. An ABL application also provides perimeter security. A single ABL application can be used to replace both classic AppServer Agents and WebSpeed Messengers. You can find details on ABL applications at ABL applications.

- One or more ABL web applications. A web application is the next level of isolation within an ABL application. It is identified by a unique URL path and security configuration. All HTTP/S requests to that URL are mapped to a published API defined as an ABL Service or are used to access static content. Each API is a logical representation and maps to specific ABL code which permits obfuscation of sensitive implementation details such as method and parameter names. Each web application is packaged as a web application archive file (.war). You can find details on ABL web applications at PAS for OpenEdge Web applications.
- One or more ABL Services. ABL Services define domain- and micro-service APIs as part of an ABL web application. The ABL Service is identified as a resource in the URL and mapped to business logic within the ABL Application. The mapping details are transport type specific—one of WEB, REST, SOAP or APSV are used. You can find details on ABL services at Create an ABL Service and ABL Service Artifacts.
- One or more Multi-session Agents (MSAgents). An MSAgent is a specialized AVM that can run multiple ABL sessions concurrently, allowing one MSAgent to handle requests from multiple PAS for OpenEdge clients. An MSAgent maps one-to-one with an OS process. Since you can run multiple ABL sessions within a single process, this highly scalable architecture uses less system resources than classic AppServer. When an MSAgent starts up, it runs the Agent Startup procedure if one was specified. This procedure can be used to perform tasks required by all server sessions that run in the agent. One common task is to create all self-service database connections that are shared by the server sessions created and managed by a given MSAgent. Each MSAgent is a user with its own self-service connection to a given database, and all its server sessions share that same connection as separate users. You can find details on Agents and ABL Sessions at Agents and Sessions and ABL Sessions.
- The Session Manager. The Session Manager is responsible for processing all incoming requests and routes them to the ABL Sessions within a MSAgent. It also manages the pool of ABL Sessions that can be running in one or more MSAgents. You can find details on the Session Manager at Broker Agent Architectures and on the session pool at ABL Session Manager and Session Pool.
- PAS for OpenEdge Clients send requests to a PAS for OpenEdge instance. PAS for OpenEdge Clients can be an ABL Client, Java Open Client, .NET Open Client, browser Client, REST clients, SOAP clients, Web UI and Mobile UI. In general, PAS for OpenEdge expects clients to operate in a stateless manner. For the purposes of supporting classic AppServer applications, the APSV and SOAP transports give you a means to emulate session-managed and session-free models. ABL Services identify the type of clients that can be supported by including a transport identifier in the URI of the service. Supported transports are:
  - **APSV** for ABL clients, Java, and .NET Open Clients
  - **WEB**, **REST** for RESTful requests
  - **SOAP** for SOAP requests
  - Static for Static content

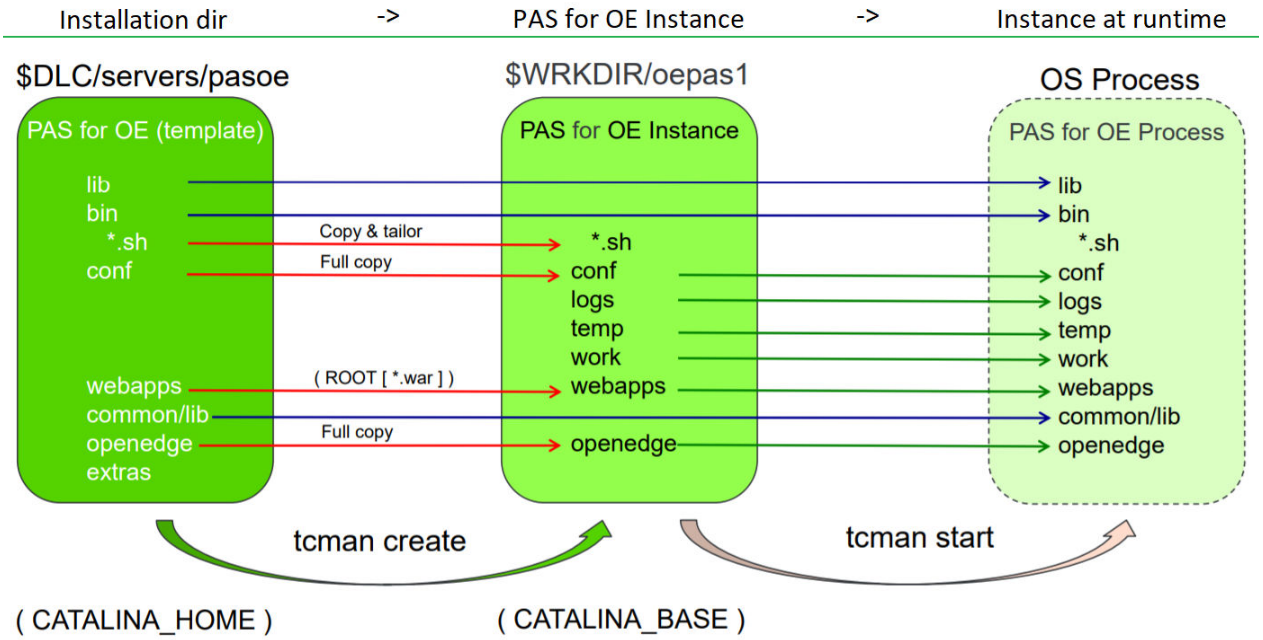
  You can find details on transports at Transports and services and details on client access migration to PAS for OpenEdge in the Application Migration and Development Guide.

For more information, see the Progress Application Server (PAS) for OpenEdge Technical Migration Guide.

Instances

Instances are a standard Apache Tomcat feature. Instances independently run copies of the PAS for OpenEdge installation. Each instance runs on its own Java Virtual Machine (JVM), has its own configuration with unique ports, and hosts its own web applications. However, each instance runs a Tomcat server that uses a number of common files from the same $CATALINA_HOME directory. Each instance has an alias. The default value is the directory in which the instance was created, but it can be set to another string.

The following figure shows the relationship between the PAS for OpenEdge installation and any PAS for OpenEdge instance that is created:



When you create an instance, the root directory of the instance is assigned to the `CATALINA_BASE` environment variable within the scripts in the `instance-path`/bin directory. The root directory of the PAS for OpenEdge installation is assigned to the `CATALINA_HOME` environment variable in the scripts in the `instance-path`/bin directory. The scope of these environment variables is limited to the context of an individual `instance-path`/bin scripts. For more information about environment variables, see Catalina environment variables.

You set the PAS for OpenEdge installation directory and PAS for OpenEdge instance directory during the OpenEdge installation, and they must be different.

The default locations are described in About licensing and installation in *Manage Progress Application Server (PAS) for OpenEdge*.

Progress requires that you deploy your web applications to an instance of PAS for OpenEdge, rather than deploying to the PAS for OpenEdge installation directory. This practice prevents the accidental corruption of the installation executables, configuration settings, and libraries. It also prevents the accidental deletion of web applications if the PAS for OpenEdge installation directory is removed during an uninstall.

Some advantages of instances are:

- Updates to the Apache Tomcat server libraries and executables do not affect your web applications. You avoid needing to update the applications or re-configuring them.
- You can establish different security policies for each of the instances.

- You can tailor the JVM for individual applications because each instance runs in its own JVM with its own configuration.
- Instances provide a quick way to create a test server for experimenting with new configurations and applications without the danger of permanently corrupting an existing server.

As shown in the previous figure, all PAS for OpenEdge instances execute a set of common JAR files, scripts, and libraries from the following directories on the parent server:

- $CATALINA_HOME/lib
- $CATALINA_HOME/common/lib
- $CATALINA_HOME/bin

Note: Catalina is the Apache product name of the component that implements the Java servlet container for Tomcat.

The structure of a PAS for OpenEdge instance (which is identified by an alias name and its OS file system path) includes the following directories:

- ablapps—A directory that lists the ABL applications deployed to the instance. Each instance is created with an ABL application of the same name listed in this directory.
- bin— A directory that contains OpenEdge and customer-written utilities and libraries that are tailored to that specific instance. For example, OpenEdge utilities are tailored to use the local file space of the instance but load and use shared libraries and utilities of the installation directory (in other words, `$CATALINA_HOME`).
- common/lib— A directory that contains Java libraries and files that are shared across all of the web applications of the instance,

  Note: The OpenEdge installation also has a common/lib directory that holds Java libraries and files shared by all web applications across all instances.

- conf— A directory that contains the Tomcat, OpenEdge, and customer-written configuration files of the instance that control only the applications deployed to the instance
- logs— A directory that contains the log files for the server instance, including all of log files for the server, web application, OpenEdge, and ABL application.
- work— A directory that contains the ABL web application-generated persistent working files. These files should not be deleted by an administrator as part of cleaning up before restarting an application.
- temp— A directory that contains the ABL web application-generated temporary files. These files can be deleted by an administrator as part of cleaning up before restarting the server instance.
- webapps| `webapps-ref` — A directory, or a reference to a directory, that contains the Tomcat, OpenEdge, and customer ABL web applications of the instance.
- openedge — A directory that contains .p and .r files, and the supporting files only for ABL applications. Other Progress and third-party web applications cannot write or read from this directory.

Additional information about instances in PAS for OpenEdge:

- An instance consists of one Apache Tomcat instance.
- An instance has a URI, for example: http://localhost:8810

- An instance has a default ABL application with the same name as the instance in the /ablapps directory.
- An instance can contain non-OpenEdge web applications (such as Progress Corticon, OpenEdge Manager, or a web UI) in the /webapps directory. OpenEdge web applications must be part of an ABL application (as specified in openedge.properties).
- An instance is the unit of deployment in a continuous integration and deployment model. Whole instances should be deployed with configurations stored in the /conf directory for all component parts. The component parts contribute their configurations up to the instance level.
- To deploy an instance, Progress recommends compressing the instance folder, copying it to the production machine, and then registering and starting the instance.

  See Package an instance for production in *Manage Progress Application Server (PAS) for OpenEdge*.
- An instance has an /openedge directory in it that is added to the PROPATH by default. This directory should contain code or libraries that are shared across all ABL applications deployed to the instance, not code for a specific ABL application.

See also:

- PAS for OpenEdge directories

ABL applications

A PAS for OpenEdge instance and the ABL applications that it hosts have a defined structure with specific locations for different components. An ABL application is the smallest unit of packaging for business application logic that can run independently on a PAS for OpenEdge instance. However, there is no direct URI to the ABL application.

An ABL Application is a logical construct within a PAS for OpenEdge instance that consists of the following:

1. One or more multi-session agents, each containing one or more AVM sessions, all configured alike.
2. A common security configuration.
3. One or more ABL web applications.
4. One or more ABL services.
5. Environment variables and lifecycle scripts.
6. Additional artifacts such as non-ABL web applications like Tomcat manager.war.

All PAS for OpenEdge instances must have at least one ABL application (abl-app) and at least one oeabl.war-based web application (abl-webapp). ABL applications are created via the following command:

```
tcman deploy webapp-name ablapp-name
```

If the `ablapp-name` is not supplied, the ABL web application will be deployed against the default ABL application. The default ABL application has the same name as the PAS for OpenEdge instance it is hosted in. All ABL applications can be found in the /ablapps directory of the instance.

An ABL application in the context of PAS for OpenEdge is a business application that has:

- A unique name that can be referenced from administrative tools.

- A set of r-code files that comprise the application.
- A single PROPATH value to access the r-code of the application.
- One or more OpenEdge database connections.
- A set of startup options.

Each ABL application can have its own configurations for agent and session settings, database connections, authentication, event procedures, and PROPATH within the same instance.

Additional information about ABL applications in PAS for OpenEdge:

- There is a single session manager for an ABL application, regardless of how many OpenEdge web applications are running on the instance.
- An ABL application must have at least one OpenEdge web application, but can have more.
- An ABL application lives in the /ablapps directory, and by default the ABL application contains a /conf directory with security information in it.

To package an ABL application for deployment to a PAS for OpenEdge instance, see OpenEdge Archive Structure in *Manage Progress Application Server (PAS) for OpenEdge*.

ABL application best practices

Even though an ABL application can physically locate its files anywhere in the OS file system, keeping them inside the PAS for OpenEdge instance `/ablapps` directory promotes scaling, extensibility, and deployment.

To promote scalability, extensibility, and deployment, an ABL application should contain the following components:

- The ABL application (identified by name) should be deployed to an existing PAS for OpenEdge instance in the `instance-name`/ablapps directory.
- Common .r or .p files used across multiple OpenEdge ABL web applications (abl-webapp) should be located in the `instance-name`/openedge/* directories. The directory structure should be capable of being configured read-only for a secure installation.
- Application-specific .r or .p files should be located in the `instance-name`/webapps/`ABL_webapp_name`/WEB-INF/openedge/* directories. The directory structure should be capable of being configured read-only for a secure installation.
- ABL web applications derived from oeabl.war can be deployed to an existing PAS for OpenEdge instance in the `instance-name`/webapps directory. They are used to host APSV, SOAP, and REST transport access to ABL business logic. For more information about ABL web applications, see Web applications.

The ROOT application

Apache Tomcat comes configured with a number of web applications, which are preconfigured in PAS for OpenEdge. Tomcat requires that there always be a ROOT application in the `instance-name`/webapps folder.

In PAS for OpenEdge, the default ROOT application is a deployment of oeabl.war, the OpenEdge web application. The file oeabl.war is deployed with every PAS for OpenEdge instance. You can undeploy ROOT, and deploy any valid web application in its place. The OpenEdge web application is a type of Java web application that is preconfigured with security and Java servlet interfaces that translate HTTP client requests into ABL requests, and schedules when they execute using the ABL session manager.

Note: The ROOT web application in a PAS for OpenEdge instance is the oeabl.war file deployed with the name ROOT, not the ROOT.war file for the Tomcat server. That file and other standard Tomcat configuration files and applications can be found in $CATALINA_HOME/extras.

When you create a PAS for OpenEdge instance, an OpenEdge ABL web application, oeabl.war, is automatically deployed in the `instance-path`/webapps/ROOT directory. The ABL web application supports all transports for an ABL application, including SOAP and REST web services, web handlers, ABL and Open Client (APSV) access. In addition, an ABL application can be represented by multiple OpenEdge ABL web applications deployed to a single PAS for OpenEdge instance.

The ROOT directory contains the manifest file for oeabl.war (in the META-INF folder), static files (in the static folder), the application definitions (in the web.xml file), and other pertinent resources such as classes, transports (APSV, REST, SOAP, and WEB), security files, and the default PROPATH folder.

Web applications

The web applications deployed to a PAS for OpenEdge instance are generated as Web Application Archive (WAR) files. When deployed, WAR files are expanded in the Java servlet container of the server in the / webapps directory of the PAS for OpenEdge instance.

In PAS for OpenEdge, the /webapps folder typically includes:

- A PAS for OpenEdge ROOT application that replaces the default Tomcat ROOT application.

  Note: The server must have an application named ROOT installed.

- (Optional) administrative applications such as manager.war and oemanager.war. For more information, see Manager applications.
- The ABL web applications, based off of oeabl.war, that the application server runs in response to a client request.

The oeabl.war application provides transports for accessing ABL and web services via HTTP. Copies of the oeabl.war application can be deployed from the `openedge-installation-dir`/servers/pasoe/extras (also known as $CATALINA_HOME/extras) directory.

As an example of a web application in the /webapps directory, the OpenEdge ABL web application (oeabl.war) is structured as follows:

- /apsv — A URI path reserved for OpenEdge client connections (can be disabled).
- /soap — A URI path reserved for OpenEdge SOAP client connections (can be disabled).
- /rest — A URI path reserved for OpenEdge REST clients (can be disabled).
- /web — A URI path reserved for OpenEdge WebSpeed and new REST clients (can be disabled).
- static — A directory that contains static files, including images, style sheets, and HTML pages.

- META-INF — A required directory that contains metadata and context.xml definitions specific to the web application. OpenEdge uses a tailored context.xml file for the web application, and can be tailored by the deployment site to meet security requirements.
- WEB-INF — A required directory that holds the configuration file of the web application (web.xml), Spring Security configuration files, and other application private files.
  - adapters — OpenEdge directory tree that hold REST, SOAP, and WEB deployment files.
  - lib — Required directory that holds any Java libraries specific to the web application.
  - classes — Required directory that holds any Java classes or data files specific to the web application.
  - openedge — Directory provided intended for ABL developer to distribute their r-code files that apply specifically to the web application.

For each ABL web application, include:

- (Optional) The .r and .p files specific to the application in the `instance-name`/webapps/`webapp-name`/WEB-INF/openedge directory. The directory structure should be capable of being configured read-only in a secure installation
- Static pages used by the web application. They should be located relative to the web application's URI from the root URI (/) , but not from one of the reserved OpenEdge URIs.
- Configuration files for the web application, including the application-specific web.xml files and oeablSecurity*.xml files in the webapps/`webappname`/WEB-INF directory.
- A default REST configuration properties file: WEB-INF/adapters/rest/runtime.props.

  Note: REST properties are not contained in the `instance-name`/conf/openedge.properties file.

- REST .paar files that may be incrementally deployed into an existing OpenEdge web application.
- SOAP .wsm files that may be incrementally deployed into an existing OpenEdge web application.
- A fragment of the openedge.properties file that contains the defaults for the ABL application and its associated web applications, transports, session manager, and multi-session agents. The fragment is appended to the `instance-name`/conf/openedge.properties file at installation time by a user-written tailoring script.
- (Optional) User-written installation tailoring script that customizes web applications and openedge.properties files.
- (Optional) User-written startup environment script (`myapp`_setenv.[sh|bat]) that sets OS process environment variables used by the ABL application code.

Additional information about ABL web applications in PAS for OpenEdge:

-
  - An ABL application contains at least one ABL web application derived from the oeabl.war application.
  - An ABL web application exposes ABL application logic using HTTP, so it can be called with a URI and with HTTP methods.
  - An ABL web application has a URI that is relative to the instance, for example, http://localhost:8810/`webapp-name`. The ROOT OpenEdge ABL web application (ROOT application) has a special relative URI to the instance, which is accessed through the instance URL, for example: http://

localhost:8810/. Therefore, if you removed the ROOT application from the /webapps folder, then the instance URL would not be accessible.

- An ABL web application can act as an authentication boundary, for example, using login web pages (JSP) or inheriting ABL application authentication.
- An ABL web application contains one or more ABL services, including a /static service.
- An ABL web application has a WEB-INF/openedge directory which is added to PROPATH when the web application is deployed. This directory contains ABL code from individual ABL services.
- Other web applications can also be deployed to an instance that are related to but not part of an ABL application, such as a monitoring application like manager.war. For more information, see Manager applications.

Transports and services

In PAS for OpenEdge, transports handle HTTP requests and are built into the default ROOT (oeabl.war) application. Transports handle requests from clients that use any one of those protocols for communication with a web server.

Transports use reserved URLs to deliver different types of web services. The supported transports and their assigned URLs are described in the following table:

Transport	Description	URL Path	Example URLs
APSV	Handles ABL clients, Java and .NET Open Clients requests.	/apsv	APSV code deployed to the ROOT application: http://localhost:8810/apsv APSV code deployed to an ABL web application other than ROOT (in this example, called `financeApp`): http://localhost:8810/financeApp/apsv **Note:** The APSV connection to PAS for OpenEdge is specified in the `-sessionModel` connection parameter argument in the ABL client application. For more information, see Connecting to a PAS for OpenEdge instance and Connection parameters argument in *Application Development with PAS for OpenEdge*.

Transport	Description	URL Path	Example URLs
REST	Handles REST requests.	/rest	REST service deployed to the ROOT application: http://localhost:8810/rest/customerSvc/Customer REST service deployed to an ABL web application other than ROOT (in this example, called `financeApp`): http://localhost:8810/financeApp/rest/customerSvc/Customer For more information about developing REST services, see Develop an ABL service using the REST transport in *Develop ABL Services*.
SOAP	Handles SOAP 1.1 requests.	/soap	SOAP service deployed to the ROOT application: http://localhost:8810/soap/ SOAP service deployed to an ABL web application other than ROOT (in this example, called `financeApp`): http://localhost:8810/financeApp/soap/ For more information about developing SOAP services, see Develop an ABL service using the SOAP transport in *Develop ABL Services*.
WEB	Handles REST, classic WebSpeed, and HTTP requests. Offers the option to modernize your application with WebHandlers.	/web	Web service deployed to the ROOT application: http://localhost:8810/web/customerSvc/Customer Web service deployed to an ABL web application other than ROOT (in this example, called `financeApp`):

Transport	Description	URL Path	Example URLs
			http://localhost:8810/financeApp/web/customerSvc/Customer The Web transport also supports using Progress Data Object (PDO) services, which are also indicated in the URI. This is shown in the following example: http://localhost:8810/web/pdo/customerSvc/Customer For more information, see the Develop a Progress Data Object Service in *Develop ABL Services*.

Note: It is not possible to remove the transport name from the URL path.

Note: Because a PAS for OpenEdge instance can contain one or more ABL web applications, if you are referencing an application other than the ROOT application, the URL path is preceded by the application name (for example, `webapp_name`/web).

ABL and web services

Additional information about ABL and web services:

- ABL services are exposed using URI.
- ABL services are grouped by transport, which is a path segment in the URI, for example, http://localhost:8810/`webapp-name`/rest.
- ABL services are represented by a .paar file (REST transport), .wsm file (SOAP transport), or a WebHandler (WEB transport). APSV services must be moved to the PROPATH.
- Transports can be secured independently. ABL services contribute authorization information to the OpenEdge ABL web application. For example, you can secure a transport URI so that it can only be accessed by an authenticated user with specific roles.
- ABL services need service interfaces (.p or .cls files). Service interfaces are intended to translate input (HTTP messages) into ABL business logic calls, and the output of those calls back into HTTP. Service interfaces can also deal with error handling or perform additional authorization.

- For more information, see Deployment artifacts and Develop ABL Services.

Manager applications

In PAS for OpenEdge, the OpenEdge Manager application, oemanager.war, is deployed to each instance in order to manage it. OpenEdge Manager is a Java web application that provides a REST API for remote administration of the OpenEdge web applications and multi-session agents. It duplicates the administration API supported by the JMX interface from Tomcat, but it uses JSON input/output payloads instead. The oemanager.war file is required to use REST APIs to manage and monitor the ABL applications and PAS for OpenEdge instances.

Tomcat Manager, which uses JMX, is deployed to a PAS for OpenEdge instance using manager.war.

Both manager.war and oemanager.war can be deployed from the $CATALINA_HOME/extras directory to an instance.

Note: Although they may not be deployed in PAS for OpenEdge, standard Tomcat configuration files and applications (like host-manager.war) are preserved in $CATALINA_HOME/extras.

Be aware that for security reasons OpenEdge Manager should not be deployed on a PAS for OpenEdge instance if, for example, you do not want remote administration enabled on a production server instance. Tomcat Manager and OpenEdge Manager are deployed on a development instance by default, and are required for integration with development tools. Some OpenEdge TCMAN commands, such as `tcman.sh info`, require Tomcat Manager to be loaded and running to work.

Catalina environment variables

There are a number of environment variables that define values for some of the key directories in PAS for OpenEdge:

- CATALINA_HOME: The root directory where you install PAS for OpenEdge. The installed server is often referred to as the PAS for OpenEdge installation directory. The CATALINA_HOME/bin directory contains all of the PAS for OpenEdge executables, including `catalina.{sh|bat}`, which is a script that is responsible for starting and stopping the server.
- CATALINA_BASE: The root directory of a run-time instance of the PAS for OpenEdge installation. If you have a number of instances, then the value of CATALINA_BASE differs for each individual instance. However, the value of CATALINA_HOME is the same for all instances of a PAS for OpenEdge installation.
- CATALINA_TMPDIR: The temporary file directory of the PAS for OpenEdge installation.
- CATALINA_PID: The path of the file holding the process ID for the Catalina Java startup process. Unlike Tomcat, PAS for OpenEdge enables the use for this variable in both UNIX and Windows systems.

Note: CATALINA_OUT is not supported in PAS for OpenEdge. To change the location of Catalina log files:

1. In ~/conf/appservers.properties, change the `catalina.logging.folder` property to the desired output path.

2. In ~/conf/openedge.properties, add the output path to the `agentLogFile` property.

PAS for OpenEdge directories

The following table lists the PAS for OpenEdge directories that are added to the standard Tomcat directory structure.

Table 1: Progress Application Server directory structure extensions

$CATALINA_HOME/ common/lib/	Contains general third-party libraries that are shared by a server, its instances, and its web applications.
$CATALINA_BASE/ common/lib/	Contains general third-party libraries that are used by a single instance and its web applications. This directory is empty by default.
$CATALINA_HOME/extras/	Contains the WAR files of the default Tomcat web applications, host-manager.war, manager.war, and ROOT.war. It can also contain the WAR files that support Progress products (for example, oeabl.war and oemanager.war).

HTTP sessions

HTTP sessions are an industry-standard feature that allow web servers to maintain user identity and store user-specific data during multiple request/response interactions between a client application and a web application.

HTTP sessions preserve:

- Information about the session itself (session identifier, creation time, time last accessed)
- Contextual information about the user (client login state, and other user information the web application needs to save)

This context information is required for load balancing. While SOAP, REST and WEB transports use HTTP sessions implicitly, the APSV transport adds this functionality through a combination of properties that are enabled by default for the ROOT application. For more information, see Enable HTTP sessions for the APSV transport.

Property files

When you create a PAS for OpenEdge instance, the properties of the instance are automatically set to certain default values. Each PAS for OpenEdge instance has its own /conf directory that contains multiple configuration files, including:

- openedge.properties—Contains the properties that must be set to create an instance. See the openedge.properties.README for more information.
- catalina.properties—Contains the Java properties used by an instance and its ABL Web applications. See the catalina.properties.README for more information.
- appserver.properties—Linked to the catalina.properties file such that any changes made to catalina.properties are dynamically reflected in appserver.properties. See the appserver.properties.README for more information.
- catalinaopts.properties—Contains name-value Java properites used by the Progress Application Server (PAS) for OpenEdge and web applications. Each non-blank and non-commented line is defined as a java system property by prefixing a `-D` and passed on from the command line to the JVM using the `CATALINA_OPTS` environment variable. See the catalinaopts.properties for more information.
- jvm.properties—Contains a list of JVM startup command-line options.
- server.xml—Contains the server features for the instance.

  Note: Progress recommends that you do not edit the server.xml file.

You can edit the property files using the PASMAN and OEPROP utilities, and OpenEdge Management.

Log files

PAS for OpenEdge creates log files at three distinct levels:

- **Tomcat logs** contain logging information for the web server and JSP servlet container of a particular PAS for OpenEdge instance. These logs include Catalina logs for the core server, localhost logs for the Tomcat virtual server, server access logs, and Tomcat Manager logs.
- **ABL application-level logs** contain logging information for ABL applications running on a PAS for OpenEdge instance. This includes Spring Security authentication and authorization process event logs, and multi-session agent logs containing logging information for the ABL Session Manager running on a PAS for OpenEdge instance.
- **Instance-level independent logs** include remote management and server support logs, and user-deployed Java web applications.

For more information, see Configure logging in PAS for OpenEdge in *Manage Progress Application Server (PAS) for OpenEdge*.

PAS for OpenEdge logs and their contents

The following tables list the log files produced by each component of PAS for OpenEdge, and the kind of information found in each log file.

The default location for all these log files is $CATALINA_BASE/logs.

The $CATALINA_BASE environment variable defines the root directory of a PAS for OpenEdge instance. On a UNIX-machine, reference the logs directory as $CATALINA_BASE/logs. On a Windows machine, reference %CATALINA_BASE%\logs.

Table 2: Tomcat logs

Filename	Description
catalina.date.log	Activity log of the Tomcat server or any of the deployed web applications.
catalina.out	A UNIX-only, catch-all log file that contains everything written to the Tomcat console output, for every application or process.
localhost.date.log	Web application activity. The default Tomcat host name log.
localhost-access.date.log	Requests processed by the web server.
manager.date.log	Activity from the Tomcat Manager.
host-manager.date.log	Activity from the Tomcat Manager for virtual hosts.

Table 3: Multi-session agent logs

Filename	Description
instance-name.agent.log	Activity from the ABL sessions, including ABL application startup and shutdown procedures.

Table 4: ABL application logs

Filename	Description
ABL-app-name.date.log	Each interaction that the ABL application has with the agent.
ABL-app-name.agent.date.log	The ABL application agent process log output.
ABL-app-name_authn.date.log	A log file that contains the audit trail of failed and passed Spring Security authentication process events. The log is shared by all deployed web apps and transports mapped to the ABL application. This log file can help detect possible server attacks, and track users who access the application.
ABL-app-name_authz.date.log	A log file that contains the audit trail of failed and passed Spring Security URL access control process events. The log is shared by all deployed web applications and transports mapped to the ABL application. This log file can help detect possible intruder scanning.

Filename	Description
oests.`date`.log	The Spring Security and Security Token Service web application log that is found only in the OpenEdge Authentication Gateway server.

Table 5: Independent web application logs

Filename	Description
oemanager.`date`.log	The log file of the remote ABL application administration API.
oehealth.`date`.log	The log file of the OpenEdge HealthScanner web application.
oedbg.`date`.log	The log file of the OpenEdge Debugger web application.

Use PAS for OpenEdge with Docker containers

You can now run PAS for OpenEdge instances in a Docker container, which means you can perform zero down-time versioning of PAS for OpenEdge instances.

Docker is a technology that enables developers to develop, deploy, and run an application in a standalone, virtualized environment called a container. Containers are a lightweight, standardized, and secure method of executing software in an isolated environment. Use containers to isolate applications for faster development and deployment.

Virtual machines offer a similar benefit as containers, but containers have the advantage because they do not virtualize hardware. Containers work on the application layer, and multiple containers share the same operating system kernel. This technology makes running multiple containers much more portable and efficient than running multiple virtual machines.

Continuous integration and continuous deployment (CI/CD) is one of the biggest benefits of using Docker containers. CI/CD enables developers to build, test, and deploy code in a wide variety of environments and enables them to test as frequently as they want. By combining the development and testing processes, CI/CD and Docker containers can make developing, testing, and deploying software much more efficient.

Containers provide several advantages when used with PAS for OpenEdge:

- Docker containers help to reduce the overhead for packaging and deploying ABL applications, enabling you to streamline your application distribution processes. You only need to create a Docker custom container image with PAS for OpenEdge once, and then you can distribute that image and run as many containers as needed.
- When you upgrade your Docker containers for PAS for OpenEdge, you only need to upgrade one image, and then you can easily redistribute the upgraded image.
- With tools such as Docker Swarm and Kubernetes, you can to run multiple PAS for OpenEdge containers in a cluster, which provides high availability and elasticity.

For more information about containers, see Learn about PAS for OpenEdge in a Docker Container in *Run PAS for OpenEdge in a Docker Container*.

PAS for OpenEdge Lite

PAS for OpenEdge Lite is a limited version of the PAS for OpenEdge production server. The two products are identical, both architecturally and functionally, but PAS for OpenEdge Lite has the following limitations compared to the PAS for OpenEdge production server:

- An instance can have only one ABL application deployed to it. There is, however, no limit to the number of instances that you can create or to the number of ABL web applications that you can deploy to an ABL application.
- Each instance can have only one multi-session agent.
- A multi-session agent can have up to five concurrent connections (requests) only. For the session-free application model, the maximum number of concurrent sessions that a multi-session agent can process is five. For the session-managed application model, there is no limit to the number of sessions that the multi-session agent can process. But, at any given point in time, a maximum of five concurrent sessions can be handled.

If your application load demands more than 5 concurrent connections, then you can set up more PAS for OpenEdge Lite instances based on the load. Though you can use the same license to set up multiple instances, you should obtain a different license for each new instance. Further, to distribute throughput across your instances, set up PAS for OpenEdge load balancing.

Note:

- During a Software Asset Management (SAM) audit, if a customer is found to have set up multiple PAS for OpenEdge Lite instances using a single paid license, the customer will be regarded as non-compliant.
- PAS for OpenEdge Lite does not include OpenEdge Authentication Gateway.

Main steps: Install and use PAS for OpenEdge

To install and use PAS for OpenEdge, developers and system administrators perform the following main steps.

Developer

As a OpenEdge developer, your responsibilities include the following:

1. **Install Progress Developer Studio.**

   A development instance of PAS for OpenEdge is installed as part of Progress Developer Studio.

2. **Create a development instance.**

Use Progress Developer Studio to create new instances, or use the default development instance, `oepas1`.

3. **Define properties for your instance.**

   Use the Progress OpenEdge options in the **Preferences** menu to set the launch configuration, startup, server, PROPATH, and database connection properties.

4. **Define security.**

   Configure the Spring Framework and authentication mechanisms.

5. **Debug your development instance.**

   Use the Debug view in Progress Developer Studio.

6. **Create archives for production.**

   Package your instance for deployment.

System Administrator

As a system administrator for PAS for OpenEdge, your responsibilities include the following steps to bring your server online:

1. **Move to a PAS for OpenEdge production instance.**
   a. **Install the license for a production instance.**

      Obtain a license for and install the production version of PAS for OpenEdge.

   b. **Create the production instance.**

      Use PASMAN or OpenEdge Management to generate a production instance.

   c. **Configure the production instance.**

      Use PASMAN or OpenEdge Management to set server properties, enable transports, and (optionally) deploy the WebSpeed WebHandlers.

   d. **Deploy applications to the production instance.**

      Use PASMAN or OpenEdge Management to deploy applications to your instance.

2. **Secure your production instance.**
   - Get certificates and keys from IT.
   - Enable TLS.
   - Check user and file permissions.
   - Set the authentication level for the instance and applications.
3. **Test and monitor the production instances.**
   - Monitor your production instance using OEJMX or REST APIs.
   - Tune your instance.

- Implement load balancing.