Progress® Application Server for OpenEdge®: Introducing PAS for OpenEdge
Copyright

© 2017 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.

Business Making Progress, Corticon, DataDirect (and design), DataDirect Cloud, DataDirect Connect, DataDirect Connect64, DataDirect XML Converters, DataDirect XQuery, Deliver More Than Expected, Icenium, Kendo UI, Making Software Work Together, NativeScript, OpenEdge, Powered by Progress, Progress, Progress Software Developers Network, Rollbase, SequeLink, Sitefinity (and Design), SpeedScript, Stylus Studio, TeamPulse, Telerik, Telerik (and Design), Test Studio, and WebSpeed are registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and/or other countries. AccelEvent, Analytics360, AppsAlive, AppServer, Arcade, BravePoint, BusinessEdge, DataDirect Spy, DataDirect SupportLink, DevCraft, DigitalFactory, Fiddler, Future Proof, High Performance Integration, JustCode, JustDecompile, JustMock, JustTrace, OpenAccess, ProDataSet, Progress Arcade, Progress Profiles, Progress Results, Progress RFID, Progress Software, ProVision, PSE Pro, SectorAlliance, Sitefinity, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, WebClient, and Who Makes Progress 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.

Please refer to the Release Notes applicable to the particular Progress product release for any third-party acknowledgements required to be provided in the documentation associated with the Progress product.

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

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

March 2017

Last updated with new content: Release 11.7.0

Updated: 2017/03/24
# Table of Contents

Preface ............................................................................................................7  
Purpose...........................................................................................................7  
Audience...........................................................................................................8  
Organization...................................................................................................8  
Typographical conventions...........................................................................8  

Chapter 1: Overview of the Progress Application Server........................11  
What is the Progress Application Server?...................................................11  
What is Progress Application Server for OpenEdge?.................................13  

Chapter 2: Apache Tomcat and the Progress Application Server for OpenEdge..................................................................................................15  
The ROOT application..................................................................................16  
The PAS for OpenEdge ROOT application....................................................16  
Transports (APSV, SOAP, REST, WEB).......................................................16  
Manager applications....................................................................................16  
Web applications............................................................................................17  
CATALINA environment variables...............................................................17  
Containers and connectors..........................................................................18  
Progress Application Server directories....................................................18  
Instances.......................................................................................................19  

Chapter 3: Comparing PAS for OpenEdge to the OpenEdge AppServer...........................................................................................................21  
Comparing the architecture of the OpenEdge AppServer and PAS for OpenEdge.................................................................22  
Properties files..............................................................................................24  
Logging............................................................................................................24  
Operating modes and session types..............................................................24  
Transports replace adapters.........................................................................25  
Broker/Agent architectures..........................................................................26  
Scaling the server..........................................................................................26  
ABL Sessions..................................................................................................27  

Chapter 4: Overview of WebSpeed Support in PAS for OpenEdge...........29  
Comparison to classic WebSpeed...................................................................30
Contents

What's new and different in WebSpeed on PAS for OpenEdge..................................................32
Migrating classic WebSpeed applications....................................................................................33
Migration notes.........................................................................................................................35

Index........................................................................................................................................37
Preface

For details, see the following topics:

• Purpose
• Audience
• Organization
• Typographical conventions

Purpose

This manual is a brief overview of the Progress® Application Server (PAS) for OpenEdge®, which supports the development and deployment of ABL Web applications that are accessed via the HTTP/HTTPS protocols. It describes how PAS for OpenEdge implements and extends the Apache Tomcat functionality upon which it is based. This manual also describes how PAS for OpenEdge differs from the "classic" OpenEdge AppServer product.

Audience

This manual is intended for ABL developers and OpenEdge application server administrators who need a high-level description of the architecture and features of the Progress Application Server for OpenEdge.
Organization

Overview of the Progress Application Server on page 11
Introduces the core Progress Application Server and the Progress Application Server for OpenEdge.

Apache Tomcat and the Progress Application Server for OpenEdge on page 15
Introduces key concepts and features of Apache Tomcat and how they are modified or extended in Progress Application Server for OpenEdge.

Comparing PAS for OpenEdge to the OpenEdge AppServer on page 21
Explains the difference between the "classic" OpenEdge AppServer and Progress Application Server for OpenEdge.

Overview of WebSpeed Support in PAS for OpenEdge on page 29
Describes WebSpeed architecture and support in PAS for OpenEdge.

Typographical conventions

This documentation uses the following typographical and syntax conventions:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bold</strong></td>
<td>Bold typeface indicates commands or characters the user types, provides emphasis, or the names of user interface elements.</td>
</tr>
<tr>
<td><em>Italic</em></td>
<td>Italic typeface indicates the title of a document, or signifies new terms.</td>
</tr>
<tr>
<td>SMALL, BOLD CAPITAL LETTERS</td>
<td>Small, bold capital letters indicate OpenEdge key functions and generic keyboard keys; for example, <strong>GET</strong> and <strong>CTRL</strong>.</td>
</tr>
<tr>
<td>KEY1+KEY2</td>
<td>A plus sign between key names indicates a simultaneous key sequence: you press and hold down the first key while pressing the second key. For example, <strong>CTRL+X</strong>.</td>
</tr>
<tr>
<td>KEY1 KEY2</td>
<td>A space between key names indicates a sequential key sequence: you press and release the first key, then press another key. For example, <strong>ESCAPE H</strong>.</td>
</tr>
</tbody>
</table>

**Syntax:**

- **Fixed width**
  - A fixed-width font is used in syntax, code examples, system output, and file names.

- **Fixed-width italics**
  - Fixed-width italics indicate variables in syntax.

- **Fixed-width bold**
  - Fixed-width bold italic indicates variables in syntax with special emphasis.

- **UPPERCASE fixed width**
  - ABL keywords in syntax and code examples are almost always shown in upper case. Although shown in uppercase, you can type ABL keywords in either uppercase or lowercase in a procedure or class.
<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period (.) or colon (:)</td>
<td>All statements except <strong>DO</strong>, <strong>FOR</strong>, <strong>FUNCTION</strong>, <strong>PROCEDURE</strong>, and <strong>REPEAT</strong> end with a period. <strong>DO</strong>, <strong>FOR</strong>, <strong>FUNCTION</strong>, <strong>PROCEDURE</strong>, and <strong>REPEAT</strong> statements can end with either a period or a colon.</td>
</tr>
<tr>
<td>[ ]</td>
<td>Large brackets indicate the items within them are optional.</td>
</tr>
<tr>
<td>[]</td>
<td>Small brackets are part of ABL.</td>
</tr>
<tr>
<td>{ }</td>
<td>Large braces indicate the items within them are required. They are used to simplify complex syntax diagrams.</td>
</tr>
<tr>
<td>{}</td>
<td>Small braces are part of ABL. For example, a called external procedure must use braces when referencing arguments passed by a calling procedure.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>Ellipses indicate repetition: you can choose one or more of the preceding items.</td>
</tr>
</tbody>
</table>
Overview of the Progress Application Server

The Progress Application Server (PAS) is a platform that provides Web server support for Progress applications. The Progress Application Server for OpenEdge is a PAS server that is specifically tailored to support ABL application development and deployment.

For details, see the following topics:

- What is the Progress Application Server?
- What is Progress Application Server for OpenEdge?

What is the Progress Application Server?

The Progress Application Server (PAS) is a platform that provides Web server support for Progress applications. Progress applications are packaged as Web application archives (WAR files) and deployed to the Java Servlet Container of a running instance of PAS. Client access to a PAS server is through HTTP/HTTPS protocols. Clients include (but are not limited to) browser-based applications and mobile applications.

The foundation of PAS is Apache Tomcat (see [http://tomcat.apache.org/](http://tomcat.apache.org/)).

Apache Tomcat is a Web server that includes a Java servlet container for hosting Web applications. The Apache Tomcat that you can download from the Apache Software Foundation is tailored primarily as a development server for testing, validating and debugging Web applications. PAS is tailored primarily as a production server for deploying Progress web applications.

PAS tailors Apache Tomcat as a production server by replacing the default Tomcat ROOT Web application with a ROOT application that:

- Returns no content if a client accesses the root URL of the server, so that information about server configuration and deployed Web applications is not accessible.
• Adds an implementation of the Spring Security framework to support user authentication and authorization

• Removes functionality, like remote administration, that could present a security risk

However, a key feature of PAS is that you can easily create and run more than one instance of the core server, and configure the instances separately to function either in a development or in a production environment. Each instance of the core PAS shares the executables and libraries of a common Tomcat server, but each instance is a separate process, running in a separate JVM, with its own configuration (ports, security framework, Web applications, etc.).

The following figure illustrates a common core PAS that supports a development server for testing, and a production server for publishing Progress Web applications.

Figure 1: The Core PAS with development and production instances
The figure shows that:

• Both the production instance and the development instance start up by invoking the same Tomcat runtime.

• Although it is possible to run the PAS core as a Web server, it usually does not run in a deployment with multiple instances. As shown here, the core is merely the source for the Tomcat executables and libraries that are common to all instances. (Among other advantages, this arrangement allows you to upgrade the core PAS without having to redeploy all of your Web applications.)

• Each instance is a unique process, running in its own JVM.

• Each instance has its own configuration, including uniquely defined ports.

• Each instance supports its own set of Web applications.

Note that each instance has a ROOT Web application tailored to its use as either a development or as a production server. Each ROOT application is also tailored to support whatever combination of Progress products that you intend to deploy. Also notice that the Development server has a manager.war application that is missing from the Production server. The manager.war application supports a variety of management and configuration utilities that you would not want to implement in a production sever.

Note: The ability to create multiple instances derived from a common core server is a powerful feature of the PAS architecture. Not only can you create and configure multiple server types, you can also create multiple servers to support load balancing. In addition, instances allow you to update the core Tomcat server without having to update or re-deploy Web applications.

See also
What is Progress Application Server for OpenEdge? on page 13

What is Progress Application Server for OpenEdge?

The Progress Application Server (PAS) is the core Web application server, based on Apache Tomcat, that is the foundation for application servers in OpenEdge as well as other Progress products. Progress Application Server for OpenEdge (PAS for OpenEdge) is a Progress Application Server that is tailored specifically to support OpenEdge applications, including WebSpeed applications.

PAS for OpenEdge is available as two separate products:

• The Progress Production Application Server for OpenEdge — configured as a secure Web server for OpenEdge application deployment

• The Progress Development Application Server for OpenEdge — configured as a Web server for developing and testing OpenEdge applications

Note: PAS for OpenEdge and the OpenEdge AppServer are two separate and unique application servers in OpenEdge. Both support ABL application development and deployment but are entirely different in their architectures.

Application development, testing, debugging, and deployment is supported by the Progress Development Studio (PDS) for OpenEdge, which includes a project type and a development server for PAS for OpenEdge. The PDS for OpenEdge is the recommended development tool for Web applications deployed on PAS for OpenEdge. For more information, see Progress Developer Studio for OpenEdge Online Help.
For information about developing new, or migrating existing ABL applications to PAS for OpenEdge, see *Progress Application Server for OpenEdge: Application Migration and Development Guide*.

Administration and configuration support is provided by the OpenEdge Management (OEM) and OpenEdge Explorer tools. For more information, see *OpenEdge Management: Progress Application Server for OpenEdge Configuration*.

PAS for OpenEdge also includes a command line utility (TCMAN), plus APIs (JMX and REST) for creating your own administrative scripts and utilities. See *Progress Application Server for OpenEdge: Administrative Guide* for more information.

**See also**

*Comparing PAS for OpenEdge to the OpenEdge AppServer* on page 21
Apache Tomcat and the Progress Application Server for OpenEdge

Progress Application Server (PAS) for OpenEdge is a Web application server based on the Apache Tomcat Web server and servlet container. This chapter introduces you to the key concepts and features of Tomcat. It also gives you an idea of how Tomcat was modified and extended in the Progress Application Server.

For more information about Tomcat, start with the Documentation Index on the Apache Website. However, be aware that PAS is an implementation of Tomcat tailored to support Progress products, and that PAS for OpenEdge is specifically tailored for ABL application development and deployment.

For details, see the following topics:

- The ROOT application
- Manager applications
- Web applications
- CATALINA environment variables
- Containers and connectors
- Progress Application Server directories
- Instances
The ROOT application

Tomcat comes configured with a number of Web applications, which are reconfigured in PAS. In particular, the ROOT application that comes with Tomcat is replaced with a safe application that does not implicitly or explicitly allow access to any part of the server or other Web applications. You may replace the ROOT application at your discretion, but Tomcat requires that there always be a ROOT application.

Note: The original Tomcat ROOT application will be packaged as a WAR file and distributed in PAS's /extras directory as root.war.

The PAS for OpenEdge ROOT application

In PAS for OpenEdge, the default ROOT application is a deployment of oeabl.war, the OpenEdge Web application. 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 their execution using the ABL Session Manager.

Note: The standard Tomcat configuration files and applications (like ROOT.war) are preserved in $CATALINA_HOME/extras.

See also

Transports (APSV, SOAP, REST, WEB) on page 16

Transports (APSV, SOAP, REST, WEB)

In Progress Application Server for OpenEdge, support for HTTP/HTTPS client access is included in the default /webapps/ROOT application. The root application contains transports that support SOAP, REST, and APSV (equivalent to AIA), and WEB (WebSpeed) access. There is no need to install and configure individual adapters for these protocols. Support for SOAP, REST, APSV, and WEB is built-in.

Manager applications

There are two Tomcat Web applications, Manager and Host Manager that provide remote online administration access. The manager application (manager.war) allows you to manage web applications, including deployment and stopping and starting web applications. In a similar way, Host Manager (host-manager.war) allows you to manage virtual hosts.

In PAS for OpenEdge, a Java Web application, oemanager.war, is deployed instead of the Tomcat Host Manager. The oemanager.war application provides a REST API for remote administration of the OpenEdge Web applications and ABL language engines. It duplicates the same administration API supported by Tomcat's JMX interface, but it uses JSON input/output payloads instead. It is required to support OEE/OEM management and monitoring of ABL applications.
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`.

Also be aware that Manager applications may not be deployed, or may be undeployed, for security reasons. For example, you probably would not want remote administration to be enabled on a production server. But note that some OpenEdge TCMAN utility commands, such as `tcman.sh info`, require the Manager application to be loaded and running. See the individual entries in the TCMAN Reference section of *Progress Application Server for OpenEdge: Administration Guide* for more information on TCMAN usage and requirements.

## Web applications

The Web applications deployed to a Progress Application Server (PAS) instance are generated as Web application archive (WAR) files. When deployed, Web application WAR files are expanded in the server's Java Servlet container (usually the server's `/webapps` directory.)

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

- A PAS for OpenEdge ROOT application that replaces the default Tomcat ROOT application. Note that the server must have an application named ROOT installed.
- Optional administrative applications.
- The ABL Web applications that the application server runs in response to a client request.

A deployed Web application is represented by a `Context` container in the `/conf/server.xml` file.

## CATALINA environment variables

There are a number of environment variables that hold paths for some of the Progress Application Server's (PAS) key directories:

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

Note: Catalina is the Apache product name of the component that implements Tomcat's Java Servlet Container.
Containers and connectors

The Progress Application Server (PAS) uses several layers of container elements to direct requests from clients to Web applications and then return the responses. The containers are configured in the server.xml file.

- **Context**: A context represents a web application and contains the path for that web application. One or more contexts are contained in a host.

- **Host**: A host element represents a virtual host and contains one or more contexts. The host associates a network name with the server. One or more hosts are contained in an engine.

- **Engine**: An engine handles the requests received from one or more connectors and returns the responses from the Web applications.

- **Service**: A service contains one or more connectors linked to a single engine. Connectors facilitate communication between an engine and web server. Connectors can use HTTP and HTTPS protocols, and they can also use Apache JServ 1.3 Protocol (AJP13), which supports load balancing.

- **Server**: A server represents a single PAS instance and contains one or more services.

The following figure shows how the containers and connectors relate to one another.

**Figure 2: PAS containers**

Progress Application Server directories

The following table lists the PAS directories added to the standard Tomcat directory structure.
Table 1: Progress Application Server directory structure extensions

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$CATALINA_HOME/common/lib/</td>
<td>Contains general 3rd party libraries that are shared by a server, its instances, and its web applications.</td>
</tr>
<tr>
<td>$CATALINA_BASE/common/lib/</td>
<td>Contains general 3rd party libraries that are used by a single instance and its web applications.</td>
</tr>
<tr>
<td>$CATALINA_HOME/extras/</td>
<td>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, which can optionally be deployed to an instance to support OpenEdge applications and management.)</td>
</tr>
</tbody>
</table>

Instances

Instances are a standard Apache Tomcat feature. They allow you to create individual deployment and/or development servers that share the core Progress Application Server that you installed.

Instances are independently running copies of the core Progress Application Server. Each instance runs on its own 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 some other string.

As a best practice, Progress recommends that you deploy your web applications to an instance of the Progress Application Server, rather than deploying to the Progress Application Server that you installed. This practice prevents accidental corruption of the core executables, configuration settings, and libraries. It also prevents accidental deletion of web applications if the core Progress Application Server is removed during an uninstall.

You use $CATALINA_HOME/bin/tcman.sh create command to create a new instance.

Some advantages of instances are:

- Updates to the core Apache Tomcat server libraries and executables do not affect your web applications. You avoid the necessity of updating the applications and/or re-configuring them.

- You can establish different security policies for each of the instances.

- You can tailor the JVM for individual applications, since each instance runs in its own JVM with its own configuration.

- Instances provide you with quick way to create a test server for experimenting with new configurations and applications without the danger of permanently corrupting an existing server.

When you create an instance, the root directory of the instance is assigned to the CATALINA_BASE environment variable within the scripts in its /bin directory. The root directory of the installed (core) Progress Application Server is assigned to the CATALINA_HOME environment variable in the scripts in the instance’s /bin directory. (Notice that the scope of these environment variables is limited to the context of an individual instance’s /bin scripts.)

All instances of a core Progress Application Server 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
Comparing PAS for OpenEdge to the OpenEdge AppServer

The PAS for OpenEdge can run the same ABL applications that run on the OpenEdge AppServer. However, PAS for OpenEdge is not an updated version of the OpenEdge AppServer. Progress Application Server for OpenEdge is a Web server based on Apache Tomcat. You install, configure, and monitor it as a Web server.

You can migrate existing ABL application code from the OpenEdge AppServer to PAS for OpenEdge fairly easily. However, you will need to package and redeploy your ABL application as a Web application.

On the client side, PAS for OpenEdge responds to requests from HTTP/HTTPS clients.

For details, see the following topics:

• Comparing the architecture of the OpenEdge AppServer and PAS for OpenEdge
• Properties files
• Logging
• Operating modes and session types
• Transports replace adapters
• Broker/Agent architectures
• Scaling the server
• ABL Sessions
Comparing the architecture of the OpenEdge AppServer and PAS for OpenEdge

The Progress Application Server for OpenEdge is an Apache Tomcat Web server and servlet container that is customized to provide application server support for OpenEdge applications and to respond to HTTP/HTTPS client requests.

If you are accustomed to working with the OpenEdge AppServer, you will find that PAS for OpenEdge is different. For example:

- The OpenEdge AppServer must be configured with an external Web server with adapters (AIA, REST, SOAP) in order to respond to HTTP/HTTPS client requests.
  
PAS for OpenEdge responds directly to HTTP/HTTPS requests because it is a Web server with a default ROOT application configured to support APSV (analogous to AIA), REST, and SOAP requests.

- Client requests handled by PAS for OpenEdge are not managed and routed by a NameServer process.

- A single PAS for OpenEdge instance supports both session-managed and session-free connection models. With OpenEdge AppServer, you must run a separate instance for each session model and operational mode that you need to support.
The following figure illustrates the difference in the overall architectures of the two servers.

Figure 3: OpenEdge AppServer compared to PAS for OpenEdge

Notice that the OpenEdge AppServer must be configured with a separate Web server in order to support HTTP/HTTPS clients. In addition, the Web server must be configured with appropriate adapters in order to communicate with the OpenEdge AppServer.

Also notice that multiple AppServer instances are required to support each operational mode, in this case state-aware and state-free. An optional NameServer directs the client request to the server running in an appropriate operational mode.

By comparison, PAS for OpenEdge is a simpler, more direct, and more efficient architecture. There is no need to install and configure a separate Web server with adapters. PAS for OpenEdge is the Web server. Moreover, a single PAS for OpenEdge instance can run sessions in both operational modes.
Properties files

Progress Application Server for OpenEdge (PAS for OE) is not a unified broker product. Unlike the OpenEdge AppServer, PAS for OE does not use a `ubroker.properties` file to hold configuration information. There is no shared properties file that can be referenced by multiple PAS server instances.

Each instance of PAS for OE has its own `/conf` directory that contains multiple configuration files, including `openedge.properties`. The `openedge.properties` file is similar in format to `ubroker.properties`. It contains some of the same properties as `ubroker.properties`, and some PAS-specific properties. Some of the unified broker properties either do not apply or operate differently in the context of PAS for OpenEdge. See the `/conf/openedge.properties.README` file for more information.

In addition to `openedge.properties`, an PAS for OpenEdge instance's `/conf` directory contains the following properties files:

- `appserver.properties` — Specifies Java properties used by PAS for OE and Web applications. See `/conf/appserver.properties.README` for more information.
- `catalina.properties` — Specifies Java system properties used by the TCMAN administration utilities and by deployed Web applications. See `/conf/catalina.properties.README` for more information.
- `jvm.properties` — Contains a list of JVM startup command line options.

See *Progress Application Server for OpenEdge: Administration* guide for more information about adding and modifying properties.

Logging

Log files for the Progress Application Server for OpenEdge are saved, by default, in an instance's `/logs` directory.

In the Classic OpenEdge AppServer, logging is enabled in the `ubroker.properties` file.

In PAS for OE, you enable logging, set logging levels, and specify the name logging files in the following properties files:

- `$CATALINA_BASE/conf/logging.properties` for Tomcat
- `$CATALINA_BASE/conf/openedge.properties` for the PAS for OpenEdge Multi-session Agent
- `$CATALINA_BASE/webapps/webapp_name/WEB-INF/logging.xml` for Web applications

Operating modes and session types

The Classic OpenEdge AppServer is configured to run in one of four operating modes (state-reset, state-aware, stateless, and state-free). Essentially, each mode specifies how a client is bound to an agent and whether context information is maintained from one client session to the next. An OpenEdge AppServer can only run in a single operating mode. You must run a separate OpenEdge AppServer for each operating mode that you need to support.

Progress Application Server for OpenEdge is simpler. Instead of a variety of operating modes, there are two session types, *session managed* and *session free*. 
Session managed in Progress Application Server for OpenEdge means that a client is bound to a single session. It is somewhat similar to the state-reset and state-aware modes in the OpenEdge AppServer with respect to the fact that context is maintained between client requests. Subsequent client requests are handled in the context of the same session.

Session free in Progress Application Server for OpenEdge means that a client is not bound to a single session. Subsequent requests from a client can be handled in different sessions.

Also, be aware that a PAS for OE instance can respond to both session managed and session free client requests. Whereas the Classic OpenEdge AppServer runs in a single operating mode and can only respond to client requests of a single operating mode type. You need to run a separate OpenEdge AppServer for each operating mode that you want to support.

Transports replace adapters

To provide HTTP/HTTPS client access to Web services associated with a classic OpenEdge AppServer, you must install WSA (SOAP) or REST adapters on a separate Web server. HTTP/HTTPS client access to OpenEdge applications that are not deployed as Web services requires the installation of an AIA adapter on a separate Web server.

In Progress Application Server for OpenEdge, support for HTTP/HTTPS Web services and OpenEdge applications is included in an instance’s default /webapps/ROOT application. The root application contains transports that support SOAP, REST, WEB, and APSV (equivalent to AIA) access. Since PAS for OpenEdge is a Web server, there is no need to install and configure adapters on a separate Web server.

Each transport, or protocol, is accessed via a default URL path in an instance. The defaults (which you can change) are listed in the following table:

<table>
<thead>
<tr>
<th>Transport</th>
<th>URL path</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>APSV</td>
<td>/apsv</td>
<td>Supports the OpenEdge AppServer protocol over HTTP</td>
</tr>
<tr>
<td>REST</td>
<td>/rest</td>
<td>Supports REST RPC over CFX and Camel</td>
</tr>
<tr>
<td>SOAP</td>
<td>/soap</td>
<td>Support SOAP 1,1</td>
</tr>
<tr>
<td>WEB</td>
<td>/web</td>
<td>Supports WebSpeed applications</td>
</tr>
</tbody>
</table>

Table 2: Default transport URL
Broker/Agent architectures

Compare the broker/agent architectures of the Classic OpenEdge AppServer and the Progress Application Server for OpenEdge that is illustrated in the following diagram:

Figure 4: Agents in Classic OpenEdge AppServer compared to PAS for OpenEdge

The architecture of the Classic OpenEdge AppServer uses a Broker to assign requests to Agents that execute as independent ABL sessions in multiple OS processes.

As the diagram shows, PAS for OpenEdge replaces the Broker with a Session Manager. The Session Manager manages a pool of ABL sessions running in the context a single Multi-session Agent (MSAgent) which is a single OS process. Note that the Session Manager can generate and manage more than one Multi-session Agent.

Scaling the server

Both the Classic OpenEdge AppServer and the Progress Application Server for OpenEdge are scalable—they can be configured to optimize performance as the workload increases. Note, the differences in how you scale each type of server in the following areas:

- Adding and trimming agents — The first step in scaling a server is to adjust the number of agents in order to maximize the server's response time under a given workload.
In both the Classic OpenEdge AppServer and PAS for OpenEdge, you can edit a properties file or use OEE/OEM to adjust the number of agents running on a server. However, PAS for OpenEdge also allows you to modify the number of sessions that an agent supports. An agent in the Classic AppServer is a single session agent, whereas PAS for OpenEdge agents are multi-session agents.

• **Load balancing** — The second step in scaling is load balancing. Load balancing basically involves adding more instances of the server and configuring an environment where the workload is share among those instances.

With the Classic OpenEdge AppServer, you can register multiple AppServer instances that support the same application services using an OpenEdge NameServer. You assign a weight factor to determine how the NameServer balances the load (client requests) among instances.

You can create multiple server instances, but there is no NameServer in PAS for OpenEdge. Instead of a NameServer, load balancing is achieved in PAS for OE through an Apache extension module or through third party applications. For example, see: http://httpd.apache.org/docs/current/mod/mod_proxy.html

### ABL Sessions

Note the following differences between the OpenEdge AppServer and the Progress Application Server (PAS) for OpenEdge with regard to ABL sessions:

• In the OpenEdge AppServer, each agent is an OS process that handles a single ABL session.

In PAS for OpenEdge, an agent is a single OS process that can run multiple ABL sessions. It is often referred to as a Multi-session Agent (MSAgent).

• The ABL Session Manager in PAS for OpenEdge is not identical to, but is functionally similar to the broker in the OpenEdge AppServer. It responds to client requests by communicating with an agent process to assign a session, to make sure that ABL code executes, and to ensure that a response is returned to the client. If there are no free sessions in one agent, it launches another agent.

**Note:** You configure properties (the number of sessions supported by an agent, the number of agent processes allowed in a server, etc.) in a PAS for OpenEdge instance’s `/conf/openedge.properties` file. It is similar in format to the `ubroker.properties` file, but it only applies to a single instance. All individual PAS for OpenEdge instances have their own `openedge.properties` file.

• Unlike the OpenEdge AppServer, you do not need to run a separate instance of PAS for OpenEdge in a mode to support a particular session model. In other words, a single instance of PAS for OpenEdge supports both session-managed and session-free connections.

The client request determines the mode of the session. Sessions running in different modes can run concurrently on a single instance of PAS for OpenEdge.

• In PAS for OpenEdge, you must specify separate startup/shutdown procedures for agents and sessions. In the Classic OpenEdge AppServer, the `srvrStartupProc` and `srvrShutdownProc` procedures execute on both agent and session start/stop events because the Classic AppServer agent is bound to a single session.
In PAS for OpenEdge, you specify startup/shutdown procedures in an instance’s 
/conf/openedge.properties file. You modify the `sessionStartupProc` and `sessionShutdownProc` 
properties to specify the procedures that execute on ABL session start/stop events. You modify the 
`agentStartupProc` and `agentShutdownProc` properties to specify the procedures that execute on 
agent start/stop events.
Overview of WebSpeed Support in PAS for OpenEdge

Progress Application Server for OpenEdge is available as a host for WebSpeed applications. You can migrate existing WebSpeed applications to PAS for OpenEdge, or you can use the Progress Developer Studio for OpenEdge to develop new WebSpeed applications that run on PAS for OpenEdge.

For details, see the following topics:

• Comparison to classic WebSpeed
• What's new and different in WebSpeed on PAS for OpenEdge
• Migrating classic WebSpeed applications
• Migration notes
Comparison to classic WebSpeed

WebSpeed on PAS for OpenEdge (PASOE) has some basic advantages over classic WebSpeed because:

- WebSpeed on PASOE employs a more integrated architecture compared to classic WebSpeed because both the Web server and the WebSpeed Transaction sever are combined in a single instance.
- WebSpeed on PASOE is more efficient than classic WebSpeed regarding the management and the availability of the agents that handle client requests.
- Classic WebSpeed only supports the GET and POST HTTP verbs. WebSpeed on PASOE supports all standard HTTP verbs.
- WebSpeed on PASOE supports event procedures that were not supported on classic WebSpeed.
- PAS for OpenEdge includes support for multiple servers in a single instance; you do not need to configure and run separate Web server, WebSpeed Transaction server, and AppServer instances.
- PAS for OpenEdge shares a single security context among the WEB transport that supports WebSpeed and the other transports (REST, SOAP, and APSV).

Architectural overview

The following figure is a simplified overview of the architectural differences between classic WebSpeed and WebSpeed on PAS for OpenEdge.
Notice that the classic WebSpeed implementation requires a Messenger application deployed on a third-party Web server to handle client requests. There maybe an optional NameServer that the Messenger consults to find a WebSpeed Transaction Server that has the resources that can satisfy the client request.
The PAS for OpenEdge instance, however, includes an Apache Tomcat Web server thereby eliminating the need for a third-party Web server, a Messenger application, and a NameServer. Instead, there is a Web Transport application to respond to client requests. The Web Transport is part of the OEABL application deployed in the Java Servlet Container of the instance.

Also note that the classic WebSpeed Broker is performs some of the same functions as the PASOE WebSpeed Session Manager. Both find an agent that can process a client request and return results. However, in classic WebSpeed agents are individual processes that the Broker starts and stops.

The Session Manager, on the other hand manages one or more multi-session agents that are instantiated on server startup. The client request is assigned to a session from a pool of readily available sessions. After the request is satisfied, the session is released and returns to the pool of available sessions. In this architecture, you avoid the overhead of having to start and stop agent processes.

What's new and different in WebSpeed on PAS for OpenEdge

These are some of the issues that you should take into account if you are migrating from classic WebSpeed to WebSpeed on PAS for OpenEdge.

- PAS for OpenEdge supports the three most common types of WebSpeed Web objects: static HTML, Embedded SpeedScript, and CGI Wrapper. It does not support HTML Mapped Web objects.

You should be able to migrate existing Web objects (other than HTML Mapped Web objects) to PAS for OpenEdge with few, if any, changes.

Basically, migration is just a matter of moving static and application files to the PAS for OpenEdge instance. Static files should be copied to the instance's /webapps/web_application/static directory. The standard location for application files is a directory under an instance's /openedge folder. Add any folders that contain WebSpeed application r-code files to the instance agent's PROPATH. See Migrating classic WebSpeed applications on page 33 and Migration notes on page 35 for more information.

- WebSpeed applications ran under a WebSpeed Transaction Server, which employs a control program, web-disp.p, that runs on all agents and executes Web objects. WebSpeed application developers often modified web-disp.p to customize the Web object execution environment.

The web-disp.p control program does not exist in PAS for OpenEdge. Web object execution is controlled by a built-in handler object, web-handler.p. This default handler can be modified to implement any web-disp.p customizations that you want to replicate. However, you cannot migrate web-disp.p to a PAS for OpenEdge instance.

- If you are creating new WebSpeed applications, you can take advantage of an ABL-based OpenHTTP object oriented layer for Web application development (similar to Java servlet APIs).

- PAS for OpenEdge supports standard HTTP verbs and status codes. Classic WebSpeed only supports GET and POST, and did not allow you to control status codes.

- Classic WebSpeed only supports a limited set of file types for uploading. WebSpeed on PAS for OpenEdge supports all media types for uploading and downloading. You can now, for example, stream binary files, PDFs, etc.

- PAS for OpenEdge does not support stateful applications in the same way they are supported in classic WebSpeed. In classic WebSpeed, a WSEU cookie instructs WebSpeed to lock an agent process to a specific client.

Instead of locking an agent process, PAS for OpenEdge maintains a stateful application by using JSESSIONID cookies. They are passed between the client and the server, and refer to session data stored on the server.
• WebSpeed Workshop is not supported for WebSpeed on PAS for OpenEdge.

• Classic WebSpeed's configuration properties are specified in a `ubroker.properties` file. On PAS for OpenEdge, WebSpeed configuration properties are specified in an instance's `/conf/openedge.properties` file.

• The classic WebSpeed utilities WSCONFIG and WTBMAN are not supported. Use the TCMAN utility for configuring and managing WebSpeed on PAS for OpenEdge.

For more information on configuring and managing WebSpeed on PAS for OpenEdge, see Progress Application Server for OpenEdge: Administration Guide. For more information on application migration and development, see Progress Application Server for OpenEdge: Application Migration and Development Guide.

Migrating classic WebSpeed applications

Migrating a WebSpeed application to a PAS for OpenEdge instance, involves moving the application's static files to a specific folder in the instance, and updating the instance's PROPATH to include the folders that contain the application's r-code.

Static files

A PAS for OpenEdge instance expects the static files that support a WebSpeed application to be in a particular location in the instance's directory structure. Static files include images and html files.

The location of static files for the default Web application is:

```
instance_name/webapps/ROOT/static
```

If you deploy another application, the default location for its static files is:

```
instance_name/webapps/webapp_name/static
```

where `webapp_name` is the name of the WebSpeed application.

r-code

To enable a PAS for OpenEdge instance to find a WebSpeed application's r-code, add any folders that contain WebSpeed application r-code files to the instance agent's PROPATH.

The default location for r-code is:

```
instance_name/openedge
```
PROPATH is set in the instance’s `../conf/openedge.properties` file. For example:

```properties

[AppServer.Agent]
agentMaxPort=62202
agentMinPort=62002
agentShutdownProc=
agentStartupProc=
agentStartupProcParam=
collectStatsData=0
flushStatsData=0
infoVersion=9010
keyAlias=
keyAliasPasswd=
keyStorePasswd=
keyStorePath=.\keys\
lockAllExtLib=
lockAllNonThreadSafeExtLib=
noSessionCache=0
numInitialSessions=5
PROPATH=${CATALINA_BASE}/openedge,$(DLC)/tty,$(DLC)/tty/netlib/OpenEdge.Net.pl

```

**Note:** `CATALINA_BASE` is an environment variable that resolves to `instance_path`. 
Migration notes

Before you migrate an existing WebSpeed application to an instance of PAS for OpenEdge, note the following:

- A PAS for OpenEdge instance must be created from an OpenEdge 11.6 or later release.
  
  For information about creating, starting and stopping instances, see Progress Application Server for OpenEdge: Administration Guide and/or OpenEdge Management: Progress Application Server for OpenEdge Configuration.

- R-code should be located on the same machine where the PAS for OpenEdge instance is running. R-code on network-mapped drives can cause issues with performance and permissions.

- Recompile r-code only if it was generated in OpenEdge 10.x or earlier releases.

- WebSpeed applications that use HTML mapping are not supported.

- WebSpeed applications with a modified web-disp.p and supporting files will not run without making changes to the default WebHandler.

- WSASP, WSISA, NSAPI, and CGIIP messengers are not necessary. They are not supported and cannot be configured in a PAS for OpenEdge instance.

  Because there is no Messenger, the application connection URL changes. In classic WebSpeed, the URL path included the Messenger in the Web server's scripts directory, and then the path to the code. For example:

  | http://hostname:port/cgi/wspd_cgi.sh/... |

  In PAS for OpenEdge, the default URL references the WebSpeed transport (wspd) and not the Messenger. For example:

  | http://hostname:port/web/... |
Progress Application Server for OpenEdge: Introducing PAS for OpenEdge