Progress®



Quick Start: Moving Your Classic AppServer Applications to the Progress® Application Server for OpenEdge®

OpenEdge®

1

About this guide

Audience

System administrators who are moving their existing OpenEdge Application Server (classic AppServer) applications to a production staging environment using the Progress Application Server for OpenEdge (PAS for OpenEdge).

Goals

Run a classic AppServer application on a PAS for OpenEdge production staging instance and update an ABL client connection to use the new APSV transport to access that application

Non-goals

Modernizing, performance tuning and securing your instance for full production requires additional steps that are not included in this document. Links are provided to additional resources in the Next steps.

Software prerequisites

To complete the steps in this guide, you will need to install the following OpenEdge 11.7.3 or higher products on Windows 64 and Linux:

- Application Server (classic AppServer)
- Progress Application Server for OpenEdge - Production (PAS for OpenEdge)
- OpenEdge RDBMS
- One of the following development licenses
- OpenEdge Studio
- Progress Development Studio for OpenEdge (PDSOE)

- 4GL Development System

Other prerequisites

To complete this guide, you will need the following permissions, assets, and skills:

- Systems permission to install and run scripts
- Access to existing classic AppServer application files
- Classic AppServer configuration skills
- Basic ABL development skills

2

Welcome

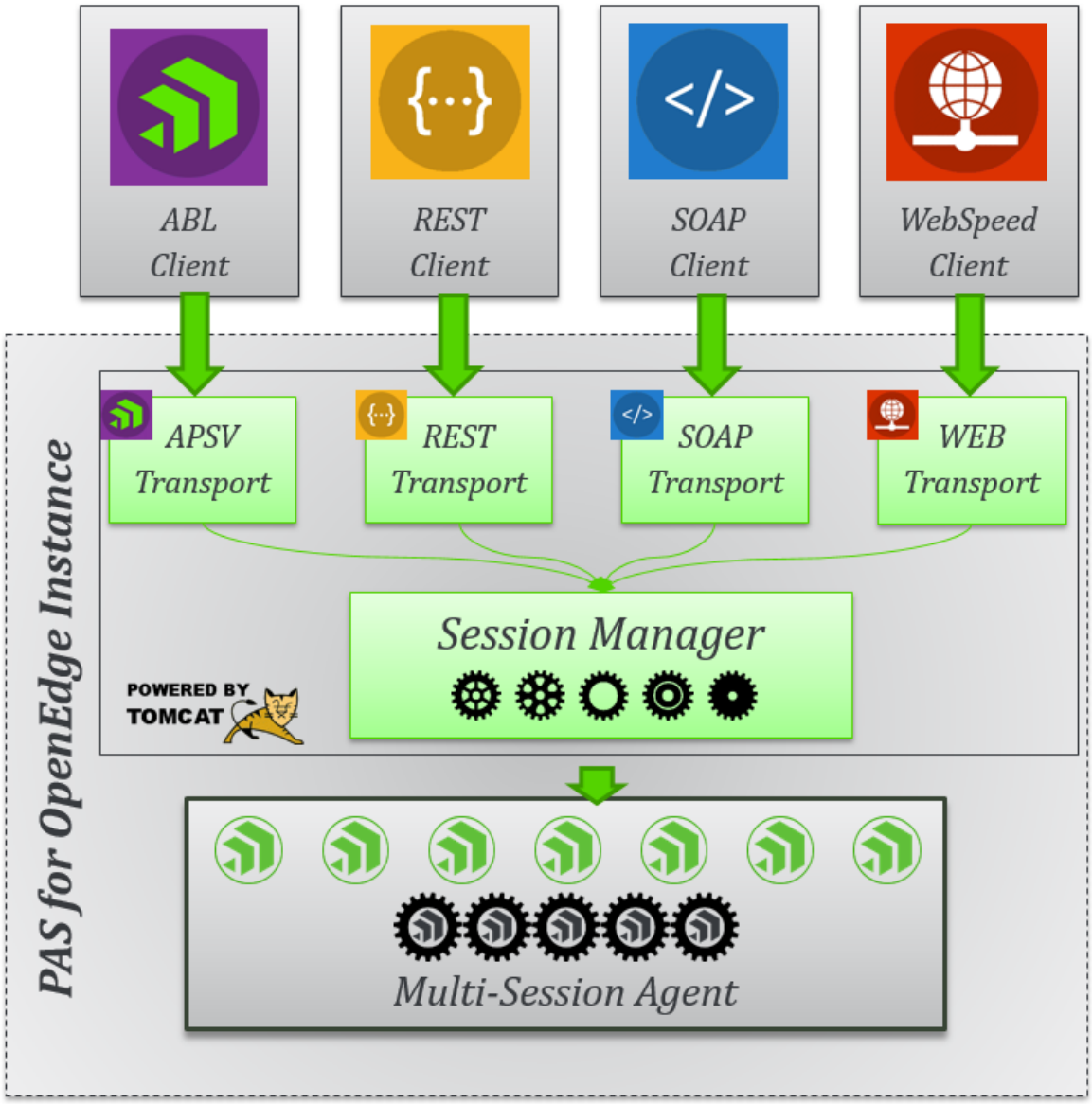
It’s moving day! As a system administrator with classic AppServer applications, it’s time for you to move those applications to a production staging instance of the Progress Application Server for OpenEdge (PAS for OpenEdge) for testing. This guide will get your applications running on the new server and your clients connected. After completing the steps in this guide, review the Next steps section, where you will find additional links for modernizing, securing and tuning your new server for production.

For details, see the following topics:

- Your new server

Your new server

PAS for OpenEdge is a **unified web server** which no longer requires special adapters. All client types send requests to your ABL server code using **one of four transports** shown below:



The **Session Manager** distributes those client requests to an underlying **Multi-Session Agent** to ensure efficient use of your server resources. Built on Apache Tomcat® web server technology and enhanced by OpenEdge development to manage the specific needs of your OpenEdge clients, you get the best of both worlds. Progress development applies Apache Tomcat updates that ensure compliance with industry standards, like **Spring security**, while supporting all client types in a single server with no additional products, using easy to configure properties files.

3

Prepare for the move

You will need to complete the following steps to setup the server and adjust its properties to support your application code:

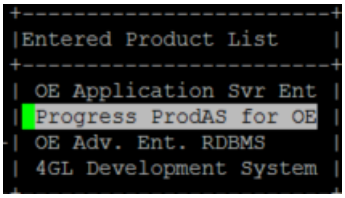
1. Install OpenEdge products
2. Create a new instance
3. Merge properties files

For details, see the following topics:

- Step 1: Install OpenEdge products
- Step 2: Create a new instance
- Step 3: Merge properties files

Step 1: Install OpenEdge products

OpenEdge offers **development** and **production** licenses for PAS for OpenEdge. For staging a production configuration, install the **production** license which implements stronger security and limits external management of your server instances. If you are less familiar with the installation process, see Unix or Windows installation steps.

1. Locate or download OpenEdge version 11.7.3 or higher. The following products were installed to complete the steps in this guide.
2. 

Step 2: Create a new instance

After installing, use `pasman create` to create an instance for testing. This instance acts as the **staging** area for deploying your existing application to a production instance before you move to a live environment. Unix commands are shown here, using a Windows command prompt will provided similar results.

1. Change to the directory where your new instance will be created and related files and logs will be stored. In this example it is the wrk directory.

```
>cd wrk
```

2. Use **proenv** to set the environment variables and paths for running commands. In this example, the installation directory is `/usr1/demo/1173`.

```
>/usr1/demo/1173/bin/proenv
```

```
$cd /usr1/demo/wrk
$/usr1/demo/1173/bin/proenv

      DLC: /usr1/demo/1173
   WRKDIR: /usr1/demo/wrk
      OEM: /usr1/demo/oemgmt
OEMWRKDIR: /usr1/demo/wrk_oemgmt

Inserting /usr1/demo/1173/bin to beginning of path and
setting the current directory to /usr1/demo/wrk.

OpenEdge Release 11.7.3 as of Mon Jan 29 18:19:06 EST 2018
```

3. Run `pasman create` to create a named instance called **myProdInstance**. (See table below for details on the options used in the example).

```
>pasman create -v -p 8817 -P 8818 -m myAdmin:myPwd -Z prod myProdInstance
```

```
$pasman create -v -p 8817 -P 8818 -m myAdmin:myPwd -Z prod myProdInstance
```

Success looks like this.

```
info: updating CATALINA_BASE in instance directory bin files
info: updating CATALINA_TMPDIR in instance directory bin files
info: updating instance directory properties
info: add instance list - myProdInstance /usr1/demo/wrk/myProdInstance
server instance myProdInstance created at /usr1/demo/wrk/myProdInstance
done - complete any manual edits at this time.
$
```

The `pasman` command manages and configures PAS instances using a variety of **actions**. The `create` action creates a new instance from a template. It takes several mandatory and optional **options**.

Option	Description
`-v`	Verbose output to console provides feedback during the create process.
`-p`	TCP port that listens for HTTP message. This example uses 8817.
`-P`	TCP port that listens for HTTPS message. This example uses 8818.
`-m`	uid:pwd is a username and password combination to replace the default tomcat:tomcat used by Apache Tomcat. The example uses myAdmin:myPwd for demonstration purposes only.
`-Z`	Identifies type of instance. The example uses prod to lock down the server with production server security settings. You may notice that development accounts and transports are disabled by default.
`instance_pathname`	Absolute path to the directory where instance gets created. The example uses myProdInstance. If no path is specified, it is created in the current directory.

For more information, see the Progress® Application Server for OpenEdge®: Administration Guide.

Run `pasman test -I instance` to confirm that the **ProtocolHandlers** initialize on the port you provided.

```
>pasman test -I myProdInstance
```

Success looks like this:

```
Jan 31, 2018 9:53:56 AM org.apache.coyote.AbstractProtocol init
INFO: Initializing ProtocolHandler ["http-nio-8817"]
Jan 31, 2018 9:53:56 AM org.apache.tomcat.util.net.NioSelectorPool getSharedSelector
INFO: Using a shared selector for servlet write/read
Jan 31, 2018 9:53:56 AM org.apache.coyote.AbstractProtocol init
INFO: Initializing ProtocolHandler ["https-jsse-nio-8818"]
Jan 31, 2018 9:53:57 AM org.apache.tomcat.util.net.NioSelectorPool getSharedSelector
INFO: Using a shared selector for servlet write/read
Jan 31, 2018 9:53:57 AM org.apache.catalina.startup.Catalina load
INFO: Initialization processed in 7744 ms
```

Option	Description
`instanceName`	Same as the instance name specified in the instance_pathname when creating.

Step 3: Merge properties files

With any move, you need to adjust to your new home. Your existing `ubroker.properties` must be converted to the new `openedge.properties` format used by your new server.

Note: Although many of the settings appear to be similar, configuring a successful deployment in the new multi-session environment is different!

In this section, you will use a command line tool, **paspropconv**, to convert properties from `ubroker.properties` and create a temporary file of changes `ubrokername.oemerge` that you can customize and then merge into the new instance's `openedge.properties`.

1. Change to the configuration conf directory to run the conversion tool in location where configuration files are stored.

```
>cd myProdInstance/conf
```

2. Run the **paspropconv** conversion tool which takes an existing `ubroker.properties` file as input and outputs a merge file with converted properties and recommendations for your new server. This tool is written in Perl and uses double hyphens (--) to run properly. For this example, the classic directory contains existing files from a classic AppServer application.

```
>paspropconv
--ubrokerPropsFile /usr1/demo/classic/ubroker.properties
--ubrokerName UBroker.AS.app_customername_prod
--pasoeAppName myProdInstance
```

Where:

`--ubrokerPropsFile`	Path to existing classic AppServer `ubroker.properties` file.
`--ubrokerName`	Fully qualified name of the broker whose properties you are converting. In this example, `UBroker.AS.app_customername_prod`.
`--pasoeAppName`	Name of the new instance. In this example, `myProdInstance`.

3. Review the `.oemerge` file, named after the `uBrokerName` you specified for the conversion.

```
$dir
app_customername_prod_setenv.bat  logging.xml
app_customername_prod_setenv.sh   myProdInstance.app_customername_prod.oemerge
appserver.properties              oeablSecurity.properties
appserver.properties.README       oeablSecurity.properties.README
catalina.policy                   oeabl_tailor.txt
catalina.properties               openedge.properties
catalina.properties.README        openedge.properties.README
context.xml                       openedge_tailor.txt
debugger.properties               PAS-default-cert.cer
instances.list                    PAS-default-cert.pem
instances.unix                    paspropconv.log
instances.windows                 server-debug.xml
jaspic-providers.xml              server.xml
jaspic-providers.xsd              tomcat-certstore.jks
jmxremote.access                  tomcat-jaas.config
jmxremote.password                tomcat-keystore.p12
jvm.properties                    tomcat-users.xml
jwtkeys                           tomcat-users.xsd
jwtRsaKeys.jks                    watcher-logging.xml
Keystore.README                   web.xml
logging.properties
```

Note: Every server configuration will be different. Read each section and adjust your environment accordingly.

Section	Description
Input arguments	Required and default arguments for paspropconv.
Operating modes	Advice on changing existing operating modes to new operating modes.
Event procedures	Advice on using new event procedures.
Manual edits	Advice on deployment difference between classic and PAS for OpenEdge systems.
Database connections	Advice on database connections.

Section	Description
Capacity & performance	Advice on properties and their impact on performance.
Redundancy & failover	Advice on minAgents and maxAgent settings.
Agent port conflicts	Advice on potential port conflicts.
Max # of ABL sessions	Advice on memory management.
Max # concurrent requests	Advice on interactions between Tomcat connector and number of concurrent requests.
SvrStartupParam	Updated property to include default PAS for OpenEdge entries.
PROPATH	Updated to use commas.
Properties to merge	Summary of the properties to merge.
Unsupported properties	Based on architectural differences between the classic AppServer and PAS for OpenEdge some properties became obsolete are no longer supported. While other properties like logging related properties are translated to Apache Tomcat implementations such as logging.xml.
Uncommented properties	The final section is the final set of properties to be merged into your new server instance's openedge.properties file.

4. Confirm or update your PROPATH entries, adding any additional or shared code locations. In this file the application code is in an appserver subdirectory.

```
#
[AppServer.Agent.myProdInstance]
    PROPATH=${CATALINA_BASE}/openedge,${CATALINA_BASE}/webapps/ROOT/WEB-INF/open
edge,${DLC}/tty,${DLC}/tty/netlib/OpenEdge.Net.pl,/usr1/demo/classic/appserver,/
usr1/demo/classic/events,/applicationname/progress/11.6/tty/rules/OpenEdge.Busin
essRules.pl,/applicationname/remote/v5_2_4,/applicationname/progress/11.6/tty/ru
les,/applicationname/progress/11.6/tty/netlib/OpenEdge.Net.pl
```

5. If you used event procedures, confirm or update your PROPATH entries, adding any additional or shared code locations.

   a. In this example, the following startup procedures are required.

```
sessionActivateProc=appsvact.p
sessionConnectProc=appsvcon.p
sessionDeactivateProc=appsvdea.p
sessionDisconnProc=appsvdis.p
sessionExecutionTimeLimit=0
sessionShutdownProc=appsvshu.p
sessionStartupProc=appsvstr.p
sessionStartupProcParam=
```

b. Confirm or update the PROPATH entries, add any additional or shared code locations.

```
#
[AppServer.Agent.myProdInstance]
    PROPATH=${CATALINA_BASE}/openedge,${CATALINA_BASE}/webapps/ROOT/WEB-INF/open
edge,${DLC}/tty,${DLC}/tty/netlib/OpenEdge.Net.pl,/usr1/demo/classic/appserver,/
usr1/demo/classic/events,/applicationname/progress/11.6/tty/rules/OpenEdge.Busin
essRules.pl,/applicationname/remote/v5_2_4,/applicationname/progress/11.6/tty/ru
les,/applicationname/progress/11.6/tty/netlib/OpenEdge.Net.pl
```

Note: PASOE does not support accessing the Windows registry. Please use environment or Java system variables.

6. If your existing application used parameter files to connect to a database or initialize other values, confirm that those files are available to the new server.

```
[AppServer.SessMgr.myProdInstance]
    agentLogEntryTypes=ASPlumbing,DB.Connects
    agentLoggingLevel=2
    agentStartupParam=-T "${catalina.base}/temp" -pf /usr1/demo/classic/startup/
databases.pf
```

7. Apply your changes, use **oeprop** to merge the converted properties into the `openedge.properties` file for your new server.

```
>oeprop
-I myProdInstance
-f myProdInstance.app_customername_prod.oemerge
```

Where:

`-I`	Instance name.
`-f`	The merge file generated by the conversion tool and reviewed by your team.

8. Copy the ubrokerName_setenv[.sh|bat] file to instancePath/bin. This script is run when the instance is started to set environment variables.

```
> cp app_customername_prod_setenv.sh
.... /usr1/demo/wrk/myProdInstance/bin
```

```
$cp app_customername_prod_setenv.sh  /usr1/demo/wrk/myProdInstance/bin
$
```

4

Move server code

You are ready to move your code.

1. Move your server code
2. Test your database access
3. Start your PAS for OpenEdge instance

For details, see the following topics:

- Step 1: Move your server code
- Step 2: Test your database access
- Step 3: Start your instance

Step 1: Move your server code

Depending on your code base, you will need to complete some or all applicable sections to move your code to the proper transport. In this guide, ABL client and application server code changes are shown.

Move ABL server code

You can run your existing code from the current location or you can take this opportunity to organize that code in the recommended application code and common code directories.

1. Move existing code to the recommended locations to promote security, scalability, extensibility, and easier deployment.
   - For application code, move compiled code to `instancePath/webapps/ROOT/WEB-INF/openedge`
   - For common code used by multiple applications, move compiled code to `instancePath/openedge`
2. Double check your `instancePath/conf/openedge.properties` and include the necessary entries.

Note: If you are moving a classic **state-reset** or **state-aware** application, you will need to configure special **connect** and **disconnect** event procedures. For more information, search the Progress® Application Server for OpenEdge®: Application Migration and Development Guide.

Move Progress WebSpeed server code

1. Move static files (images and HTML pages), from your WebSpeed application to `instancePath/webapps/ROOT/static`.
2. Review the PROPATH entry in `instancePath/conf/openedge.properties` to make sure that it includes compiled code for your WebSpeed application.
3. Edit `openedge.properties` to set the **defaultHandler** property to `OpenEdge.Web.CompatibilityHandler`.
4. If you have modified your `web-disp.p`, you will need to make similar changes to the default `web-handler.p`.
5. Reminder, OpenEdge does not support applications with HTML Mapped Web Objects.

For more information on WebSpeed code, see the Progress Application Server for OpenEdge®: Administration Guide.

Move REST server code

A classic REST web application **.war** file cannot be deployed to PAS for OpenEdge. If you have a .war, unzip the archive and deploy the application **.paar** file or export the file from Progress Developer Studio for OpenEdge.

1. Copy the code that supports the REST interface API into the directory: `instancePath/webapps/ROOT/WEB-INF/openedge`
2. Navigate to `instancePath/bin`.
3. Run `deployREST[.sh|.bat] source_descriptor ROOT`

   Where:

`source_descriptor`	Specify the path of the source descriptor, which can be either a **.paar** file containing the descriptor for the REST service or a ZIP file containing Mobile catalog files (or other static files).
`service_name`	Specify the target service name.

Note: You **do not** deploy the REST Manager (oerm.war) with PAS for OpenEdge, since the supporting files are already included in the server.

For more information on deployREST, search the Progress Application Server for OpenEdge®: Administration Guide.

Move SOAP server code

The Web Service Adapters (WSA) are no longer required for SOAP clients. The server includes the necessary SOAP transport support.

1. Copy the code that supports the SOAP interface API into one of the PROPATH locations.
2. Navigate to `instancePath/bin`.
3. Run `deploySOAP[.sh|.bat] source_descriptor ROOT`

   Where:

`source_descriptor`	Specify the path of the source descriptor, which is a WSM file.
`service_name`	Specify the target service name.

For more information on deploySOAP, see the Progress Application Server for OpenEdge®: Administration Guide.

Step 2: Test your database access

1. Start the database server, if it isn't already running.

```
>proserve /usr1/demo/classic/db/sports
```

```
$proserve /usr1/demo/classic/db/sports
OpenEdge Release 11.7.3 as of Thu Feb  1 18:18:58 EST 2018
08:58:56 BROKER     The startup of this database requires 17Mb of shared memory.
  Maximum segment size is 1024Mb.
08:58:56 BROKER  0: Multi-user session begin. (333)
08:58:56 BROKER  0: Before Image Log Initialization at block 0  offset 620. (153
21)
08:58:56 BROKER  0: Login by root on /dev/pts/57. (452)
$
```

2. Confirm that the recommended properties are refencing your database server. You will want to use a fully qualified path name for your database server.

```
>more /usr1/demo/classic/startup/database.pf
```

```
$more /usr1/demo/classic/startup/databases.pf
-db /usr1/demo/classic/db/sports
$
```

Step 3: Start your instance

Before attempting to test client connections, it is helpful to test server startup procedure and database connectivity. In this step, restart the server to confirm that the startup procedures were available and the .pf file connections are working.

1. To start the instance, run **pasman pasoestart**. This will load your new `openedge.properties` files enabling the APSV transport on your production instance, a setting which is disabled by default on a new production instance.

```
> pasman pasoestart -restart -I myProdInstance
```

Where:

`-restart`	If the instance is already running or is in a hung state, attempt to stop it, and then execute a full start up. If the instance is already in the stopped state, this option has no effect.
`-I`	Specify the instance name.

```
$pasman pasoestart -restart -I myProdInstance
Starting stopped PASOE instance myProdInstance
........

Start action: start
Initial state: stopped
Initial processes: 0
Exit state: started
Exit description: Starting stopped PASOE instance myProdInstance
Exit processes: 108182 108328
Exit status: 0
Exit errors:
$
```

Troubleshooting tips:

- Review logs in the instance/logs directory for startup errors
- Recheck the recommendations in openedge.properties
- Confirm .pf file has the correct path to the database
- Confirm that event procedures are available in the PROPATH

5

Connect clients with new transports

To connect client applications with new transports:

1. Update your client connections.
2. Test your client connections.
3. Stop your instance.

For details, see the following topics:

- Step 1: Update your client connections
- Step 2: Test your client connections
- Step 3: Stop your instance

Step 1: Update your client connections

All client connections must be updated to use one of the four transports describe in the Your new server section. This table summarizes those updates.

Application changes by client types…
AIA

Application changes by client types…
From: -URL http://host:port/aia/Aia?AppService=brokername To: -URL http://host:port/apsv
OpenEdge
From: -S port -H host -AppService brokername To: -URL http://host:port/apsv
REST
From: http://host:port/restservice/rest/restresource To: http://host:port/rest/restresource
AVM

Application changes by client types…

From:

```
-WSDL http://host:port/wsa/wsa1/wsdl?targetURI=urn:CustomerSvc
```

To:

```
-WSDL http://host:port/soap/wsdl?targetURI=urn:CustomerSvc
```

From:

```
http://host:port/cgi/wspd_cgi.sh/ . . .
```

To:

```
http://host:port/web/ . . .
```

WSASP, WSISA, NSAPI, and CGIIP messengers are not used with PAS for OpenEdge.

1. For this example, we'll update an OpenEdge ABL client connection. Open the file in the editor.

```
>mpro /usr1/demo/classic/client/ClassicClient.p
```

2. Replace the classic connections parameters with updated connections parameters.

```
File Edit Search Buffer Compile Tools Help
/* Hardcode the customer number for a demo */
ASSIGN iCustNum=10.

CREATE SERVER hServer.

/* edit made to connect could be in parameter file for easier reuse */
cConnect = "-URL http://machinename:8817/apsv -sessionModel Session-Free".
lReturn = hServer:CONNECT(cConnect).

IF (lReturn) THEN DO:
   RUN ServerGetCustNameSample.p ON hServer (INPUT iCustNum, OUTPUT cCustName).
   DISPLAY "Customer Name: " cCustName FORMAT "x(40)" SKIP.
   hServer:DISCONNECT().
END.

OUTPUT CLOSE.

QUIT.
- File: /usr1/demo/wrk/PASforOEClient.p
```

3. Save changes.

4. Recompile the client code.
5. No changes are required on the server code.

```
File Edit Search Buffer Compile Tools Help

/* Sample code, run on Appserver, gets customer's name
   from "sports" database, based on customer number
   sent from client.
*/

DEF INPUT PARAM customerNumber AS INTEGER.
DEF OUTPUT PARAM customerName AS CHAR.

IF CONNECTED("sports") THEN DO:
      FIND FIRST customer WHERE custNum = customerNumber NO-LOCK NO-ERROR.
      IF AVAILABLE customer THEN
          customerName = Name.
      ELSE
          customerName = "No record".
/*          PUT UNFORMATTED "CustomerName = " customerName SKIP. */
END.
ELSE
   customerName = "Not connected to sports".
```

Step 2: Test your client connections

The final test is to run a client connection that collects data from your database.

1. Run the code. Verify that your application can connect and access your database data.

```
                cCustName
                ---------
Customer Name:  Just Joggers Limited
```

Step 3: Stop your instance

If you are done testing your server and it is no longer needed, it is good practice to stop the instance.

1. Use the pasman command with the stop action to stop a running instance.

```
>pasman stop -I myProdInstance
```

```
$pwd
/usr1/demo/wrk/myProdInstance/conf
$pasman stop -I myProdInstance
Using CATALINA_BASE:   /usr1/demo/wrk/myProdInstance
Using CATALINA_HOME:   /usr1/demo/1173/servers/pasoe
Using CATALINA_TMPDIR: /usr1/demo/wrk/myProdInstance/temp
Using JRE_HOME:        /usr1/demo/1173/jdk
Using CLASSPATH:       /usr1/demo/1173/servers/pasoe/bin/bootstrap.jar:/usr1/dem
o/1173/servers/pasoe/bin/tomcat-juli.jar
Using CATALINA_PID:    /usr1/demo/wrk/myProdInstance/temp/catalina-myProdInstanc
e.pid
Tomcat stopped.
```

To prepare the test instance for full production release, continue your learning with the content provided in the following sections:

- Important details about your new server
- Next Steps

6

Important details about your new server

Here are a few important details about your new server:

PAS for OpenEdge differs fundamentally from the classic AppServer

Although both PAS for OpenEdge and the classic AppServer run ABL business applications, the architecture and configuration are fundamentally different. PAS for OpenEdge is an application server based upon a **web server** that uses special web applications to run ABL code. It has all the behavior and features of a web server. Although many of the configuration parameters appear to be similar, the tools and techniques for a successful deployment in the multi-session environment are different. Refer to the output of the property conversion tool for useful information on configuring your instance.

The latest release keeps you up-to-date with security and software updates

In the web environment, **tracking the latest security changes is extremely important**. The PAS for OpenEdge product is continually being updated with new functionality, bug-fixes, and security patches. Stay current to keep your application secure and stable.

Use property files to change PAS for OpenEdge configuration values

Although PAS for OpenEdge is based on Apache Tomcat®, PAS for OpenEdge has simplified the underlying Tomcat configuration and startup for local and remote administration. Rather than making changes to the `conf/server.xml` or `bin/setenv.[bat|sh]` files directly, PAS for OpenEdge uses **property files** and customer extensible script files. See Property files

Check all PAS for OpenEdge log files for errors

A client request traverses multiple PAS for OpenEdge subsystems, any one of which can raise an execution error. The recording of the error will occur in the subsystem specific log file. Always check all log files in the logs directory when investigating execution errors. You will find the log files in the `instancePath/logs` directory. For more information, see the Log files

PAS for OpenEdge may fail to start because of errors in your ABL application

The PAS for OpenEdge multi-session agent log file does distinguish between ERROR, WARNING, or INFO messages. As a result, the PAS for OpenEdge pasoestart command can only report whether the agent OS process has stopped due to startup problems. If pasoestart does not specifically report the startup error, you should manually inspect the agent log file in the instancePath/logs directory to determine what the startup problem was and correct it.

Configure and load test your instance before moving into production

Choosing the right machine image size and optimal PAS for OpenEdge configuration for that machine image plays a key role in moving your ABL application into the production environment. PAS for OpenEdge is optimized for resource consumption and optimal throughput under load. Unlike with classic AppServer or WebSpeed you cannot derive ABL application performance using a single client. Make sure to test your application with the anticipated number of concurrent clients to ensure you do not exceed finite resource limits imposed by the OS, OS processes, networking, and OpenEdge database. For more information, see Goals and common steps for tuning PAS for OpenEdge instances

Stopping a PAS for OpenEdge instance may take some time

Stopping a PAS for OpenEdge instance may take several minutes in some cases. It may even appear hung. PAS for OpenEdge (Tomcat) has a policy that allows its web applications to finish client requests before stopping. A normal PAS for OpenEdge stop is more of a suggestion rather than a hard stop of all processes. If you want PAS for OpenEdge not to wait for client requests to finish before stopping use the pasman stop command and supply the -F option to force shutdown.

The Apache Tomcat® access log is a great troubleshooting tool

The access log is a great help when troubleshooting client connection problems, maximum response times, client login failures from hackers, and much more. While it does take some small amount of processing time and disk space, its information cannot be easily obtained from other sources and can be helpful in monitoring the health of your server and tuning its run-time. See the Apache Tomcat® Configuration Reference.

Apache Tomcat® stuck thread detection is a useful alerting mechanism

The Apache Tomcat® stuck thread detection identifies requests that take a long time to process, might indicate that something is wrong in the request execution. Stuck thread detection is an alerting mechanism that logs a message for requests that take longer than a configurable amount of time to complete. See the Apache Tomcat® Configuration Reference.

UNIX file permissions that only allow the root use

On UNIX, the PAS for OpenEdge production installation and instance tailoring eliminates all access by any user account. After an OpenEdge installation completes, its instances are configured to be started, stopped, configured, and monitored by the root user account (required by the OpenEdge installer). If all your OS administration is performed via the root user account, no action needs to take place. If you follow best practices and never use the OS root user account, then you will need to change file permissions for core and instance executables and scripts. See the User and file permissions.