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** .........................................................................................................................11
  - Purpose..................................................................................................................................11
  - Audience ............................................................................................................................11
  - Organization .......................................................................................................................12
  - Using ABL documentation .................................................................................................12
    - References to ABL compiler and run-time features ..................................................13
    - References to ABL data types .......................................................................................13
  - Typographical conventions ................................................................................................13
  - Examples of syntax descriptions .....................................................................................14
    - Long syntax descriptions split across lines ..............................................................16
    - Complex syntax descriptions with both required and optional elements ................16
  - OpenEdge messages .........................................................................................................17
    - Obtaining more information about OpenEdge messages .........................................17

**Chapter 1: Overview of PAS for OpenEdge administration** ......................................19
  - Development servers compared to production servers ...............................................20
  - Managing the ROOT Web application ...........................................................................20
  - Transport URLs .................................................................................................................20
    - Modifying the WEB transport URL ........................................................................21
  - Tomcat logging ..................................................................................................................23
  - Configuration and properties files ..................................................................................24
  - Progress Application Server properties .......................................................................25
  - Progress Application Server directories .....................................................................33

**Chapter 2: Administrative tools** ................................................................................35
  - Tomcat utilities ..................................................................................................................35
  - TCMAN ...............................................................................................................................36
  - PASOESTART .....................................................................................................................37
  - PASMAN ............................................................................................................................43
  - OEPROP ...............................................................................................................................45
  - Utilities for managing keys and digital certificates ......................................................48
  - OpenEdge Management and OpenEdge Explorer .......................................................48
  - JConsole and JMX .............................................................................................................49
    - Running JConsole locally ............................................................................................50
    - Running JConsole remotely .........................................................................................50

**Chapter 3: Working with Instances** ..........................................................................53
Chapter 8: Web application security configurations

Spring Security configuration files
Security properties files
URL access control file
Enabling security for REST, WEB, APSV and SOAP
User account configurations
Local and Extended Local user accounts
LDAP user accounts
Active Directory user accounts
OERealm user account configurations
Multi-domain support for user accounts
Login models
CORS support
Adding CORS support
Single sign-on (SSO) support in PAS for OpenEdge

Chapter 9: WebSpeed configuration and management

Tools for managing and configuring WebSpeed on PAS for OpenEdge
Configuring openedge.properties
Security
Migrating classic WebSpeed applications

Chapter 10: TCMAN Reference

Using TCMAN
The tcman command
Extending TCMAN
Manager actions
List deployed applications (list)
Display OS and server information (info)
Deploy a Web application (deploy)
Undeploy a Web application (undeploy)
Reload a Web application (reload)
Display detailed server status (status)
Display memory leaks (leaks)
Start a Web application (enable)
Stop a Web application (disable)
Display global server resources (resources)
Display Web application HTTP sessions (sessions)
Server actions
Create an instance (create)
Delete an instance (delete)
Display and manage an instance's configuration (config)
Chapter 11: REST API Reference for oemanager.war.................................159

Session management..................................................................................159
Get session information........................................................................160
Get client information.........................................................................161
Get current request information.........................................................162
Get configuration properties..............................................................163
Update configuration properties.........................................................165
Get runtime metrics...........................................................................166
Reset runtime metrics.......................................................................167
Cancel a request...............................................................................168
Terminate a session...........................................................................169
Agent management..............................................................................170
List agents.........................................................................................170
Get agent status information............................................................171
Get session metrics.........................................................................172
Get connection information...............................................................173
Get request information................................................................174
Get configuration properties...............................................................178
Update configuration properties.........................................................179
Stop an agent....................................................................................180
Transport management......................................................................181
Get transport configuration properties (SOAP/REST)......................182
Update properties of a transport (SOAP/REST)..............................183
Enable or disable a transport (SOAP/REST)......................................185
Enable or disable a transport (APSV)..............................................186
Get runtime metrics of a transport (SOAP/REST).........................187
Get runtime metrics of a transport (APSV).......................................188
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset runtime metrics for a transport (SOAP/REST)</td>
<td>189</td>
</tr>
<tr>
<td>Reset runtime metrics of a transport (APSV)</td>
<td>190</td>
</tr>
<tr>
<td>List deployed services (SOAP/REST)</td>
<td>191</td>
</tr>
<tr>
<td>Get information about a service (SOAP/REST)</td>
<td>193</td>
</tr>
<tr>
<td>Get configuration properties of a service (SOAP)</td>
<td>194</td>
</tr>
<tr>
<td>Deploy a new service (REST)</td>
<td>195</td>
</tr>
<tr>
<td>Deploy a new service (SOAP)</td>
<td>196</td>
</tr>
<tr>
<td>Update runtime properties of a service (SOAP)</td>
<td>197</td>
</tr>
<tr>
<td>Get runtime metrics for a service (SOAP)</td>
<td>198</td>
</tr>
<tr>
<td>Reset runtime metrics for a service (SOAP)</td>
<td>200</td>
</tr>
<tr>
<td>Enable or disable a service (SOAP)</td>
<td>200</td>
</tr>
<tr>
<td>Remove a service (SOAP/REST)</td>
<td>201</td>
</tr>
<tr>
<td>Application management</td>
<td>203</td>
</tr>
<tr>
<td>List ABL application information</td>
<td>203</td>
</tr>
<tr>
<td>Get information about a specific ABL application</td>
<td>205</td>
</tr>
<tr>
<td>List Web applications deployed to an ABL application</td>
<td>207</td>
</tr>
<tr>
<td>Get information about a specific Web application</td>
<td>209</td>
</tr>
</tbody>
</table>
Preface

For details, see the following topics:

- Purpose
- Audience
- Organization
- Using ABL documentation
- Typographical conventions
- Examples of syntax descriptions
- OpenEdge messages

Purpose

This is a guide to administering Progress Application Server for OpenEdge that focuses on the use of command line utilities (TCMAN) and management APIs (oemanager). For information on administration with OpenEdge Management or OpenEdge Explorer, see OpenEdge Management: Progress Application Server for OpenEdge Configuration.

Audience

This manual is intended for any OpenEdge application server administrators or ABL developers who need to create, configure, or administer instances of the Progress Application Server for OpenEdge.
Organization

Overview of PAS for OpenEdge administration on page 19
Introduces the administrative tools and concepts for Progress Application Server (PAS) for OpenEdge.

Working with Instances on page 53
Explains how to create and manage instances of the core PAS for OpenEdge server.

Application deployment and management on page 61
Explains how to deploy, manage, and package PAS for OpenEdge Web applications.

HTTP sessions on page 65
An overview of HTTP sessions which are a standard method for maintaining context during a series of requests/responses between an HTTP client and a Web application.

Load balancing options for PAS for OpenEdge on page 69
Describes three alternatives for configuring load balancing among PAS for OpenEdge instances.

Server security on page 79
Provides an overview of security features and concepts in PAS for OpenEdge.

Web application security configurations on page 91
Explains how to implement Spring Security models, authentication and authorization protocols, and single sign-on for Web applications deployed to a PAS for OpenEdge instance.

WebSpeed configuration and management on page 111
Describes configuration of PAS for OpenEdge for deployment of WebSpeed applications.

TCMAN Reference on page 115
A command reference for the TCMAN command line management utility.

REST API Reference for oemanager.war on page 159
An API reference describing the REST services that can be used to manage a PAS for OpenEdge instance.

Note: For additional PAS for OpenEdge documentation, including white papers on advanced topics, see:
https://community.progress.com/community_groups/openedge_development/m/documents

Using ABL documentation

OpenEdge provides a special purpose programming language for building business applications. In the documentation, the formal name for this language is ABL (Advanced Business Language). With few exceptions, all keywords of the language appear in all UPPERCASE, using a font that is appropriate to the context. All other alphabetic language content appears in mixed case.

For the latest documentation updates see the OpenEdge Product Documentation Overview page on Progress Communities:
References to ABL compiler and run-time features

ABL is both a compiled and an interpreted language that executes in a run-time engine. The documentation refers to this run-time engine as the ABL Virtual Machine (AVM). When the documentation refers to ABL source code compilation, it specifies ABL or the compiler as the actor that manages compile-time features of the language. When the documentation refers to run-time behavior in an executing ABL program, it specifies the AVM as the actor that manages the specified run-time behavior in the program.

For example, these sentences refer to the ABL compiler’s allowance for parameter passing and the AVM’s possible response to that parameter passing at run time: “ABL allows you to pass a dynamic temp-table handle as a static temp-table parameter of a method. However, if at run time the passed dynamic temp-table schema does not match the schema of the static temp-table parameter, the AVM raises an error.” The following sentence refers to run-time actions that the AVM can perform using a particular ABL feature: “The ABL socket object handle allows the AVM to connect with other ABL and non-ABL sessions using TCP/IP sockets.”

References to ABL data types

ABL provides built-in data types, built-in class data types, and user-defined class data types. References to built-in data types follow these rules:

• Like most other keywords, references to specific built-in data types appear in all UPPERCASE, using a font that is appropriate to the context. No uppercase reference ever includes or implies any data type other than itself.

• Wherever integer appears, this is a reference to the INTEGER or INT64 data type.

• Wherever character appears, this is a reference to the CHARACTER, LONGCHAR, or CLOB data type.

• Wherever decimal appears, this is a reference to the DECIMAL data type.

• Wherever numeric appears, this is a reference to the INTEGER, INT64, or DECIMAL data type.

References to built-in class data types appear in mixed case with initial caps, for example, Progress.Lang.Object. References to user-defined class data types appear in mixed case, as specified for a given application example.

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>Bold</td>
<td>Bold typeface indicates commands or characters the user types, provides emphasis, or the names of user interface elements.</td>
</tr>
<tr>
<td>Italic</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, GET and CTRL.</td>
</tr>
<tr>
<td>Convention</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>KEY1+KEY2</td>
<td>A plus sign between key names indicates a <strong>simultaneous</strong> key sequence: you press and hold down the first key while pressing the second key. For example, CTRL+X.</td>
</tr>
<tr>
<td>KEY1 KEY2</td>
<td>A space between key names indicates a <strong>sequential</strong> key sequence: you press and release the first key, then press another key. For example, ESCAPE H.</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. |
| Period (.) or colon (:)| All statements except **DO**, **FOR**, **FUNCTION**, **PROCEDURE**, **and** **REPEAT** **end with a period.** **DO**, **FOR**, **FUNCTION**, **PROCEDURE**, **and** **REPEAT** statements can end with either a period or a colon. |
| [ ]             | Large brackets indicate the items within them are optional.                                                                               |
| [ ]             | Small brackets are part of ABL.                                                                                                           |
| { }             | Large braces indicate the items within them are required. They are used to simplify complex syntax diagrams.                             |
| {}              | Small braces are part of ABL. For example, a called external procedure must use braces when referencing arguments passed by a calling procedure. |
| |               | A vertical bar indicates a choice.                                                                                                        |
| . . .            | Ellipses indicate repetition: you can choose one or more of the preceding items.                                                        |

**Examples of syntax descriptions**

In this example, **ACCUM** is a keyword, and **aggregate** and **expression** are variables:
Syntax

`ACCUM aggregate expression`

`FOR` is one of the statements that can end with either a period or a colon, as in this example:

```
FOR EACH Customer NO-LOCK:
   DISPLAY Customer.Name.
END.
```

In this example, `STREAM stream`, `UNLESS-HIDDEN`, and `NO-ERROR` are optional:

Syntax

```
DISPLAY [ STREAM stream ] [ UNLESS-HIDDEN ] [ NO-ERROR ]
```

In this example, the outer (small) brackets are part of the language, and the inner (large) brackets denote an optional item:

Syntax

```
INITIAL [ constant [, constant ] ]
```

A called external procedure must use braces when referencing compile-time arguments passed by a calling procedure, as shown in this example:

Syntax

```
{ &argument-name }
```

In this example, `EACH`, `FIRST`, and `LAST` are optional, but you can choose only one of them:

Syntax

```
PRESELECT [ EACH | FIRST | LAST ] record-phrase
```

In this example, you must include two expressions, and optionally you can include more. Multiple expressions are separated by commas:
Syntax

```
MAXIMUM ( expression , expression [ , expression ] ... )
```

In this example, you must specify MESSAGE and at least one expression or SKIP [( n )], and any number of additional expression or SKIP [( n )] is allowed:

```
MESSAGE { expression | SKIP [ ( n ) ] } ...
```

In this example, you must specify {include-file, then optionally any number of argument or &argument-name = "argument-value", and then terminate with }:

```
{ include-file
  [ argument | &argument-name = "argument-value" ] ... }
```

Long syntax descriptions split across lines

Some syntax descriptions are too long to fit on one line. When syntax descriptions are split across multiple lines, groups of optional and groups of required items are kept together in the required order.

In this example, WITH is followed by six optional items:

```
WITH [ ACCUM max-length ] [ expression DOWN ]
  [ CENTERED ] [ n COLUMNS ] [ SIDE-LABELS ]
  [ STREAM-IO ]
```

Complex syntax descriptions with both required and optional elements

Some syntax descriptions are too complex to distinguish required and optional elements by bracketing only the optional elements. For such syntax, the descriptions include both braces (for required elements) and brackets (for optional elements).

In this example, ASSIGN requires either one or more field entries or one record. Options available with field or record are grouped with braces and brackets:
Syntax

```plaintext
ASSIGN { [ FRAME frame ] { field [ = expression ] } 
    [ WHEN expression ] } . . .
| { record [ EXCEPT field . . . ] }
```

OpenEdge messages

OpenEdge displays several types of messages to inform you of routine and unusual occurrences:

- **Execution messages** inform you of errors encountered while OpenEdge is running a procedure; for example, if OpenEdge cannot find a record with a specified index field value.

- **Compile messages** inform you of errors found while OpenEdge is reading and analyzing a procedure before running it; for example, if a procedure references a table name that is not defined in the database.

- **Startup messages** inform you of unusual conditions detected while OpenEdge is getting ready to execute; for example, if you entered an invalid startup parameter.

After displaying a message, OpenEdge proceeds in one of several ways:

- Continues execution, subject to the error-processing actions that you specify or that are assumed as part of the procedure. This is the most common action taken after execution messages.

- Returns to the Procedure Editor, so you can correct an error in a procedure. This is the usual action taken after compiler messages.

- Halts processing of a procedure and returns immediately to the Procedure Editor. This does not happen often.

- Terminates the current session.

OpenEdge messages end with a message number in parentheses. In this example, the message number is 200:

```
** Unknown table name table. (200)
```

If you encounter an error that terminates OpenEdge, note the message number before restarting.

Obtaining more information about OpenEdge messages

In Windows platforms, use OpenEdge online help to obtain more information about OpenEdge messages. Many OpenEdge tools include the following Help menu options to provide information about messages:

- Choose **Help > Recent Messages** to display detailed descriptions of the most recent OpenEdge message and all other messages returned in the current session.

- Choose **Help > Messages** and then type the message number to display a description of a specific OpenEdge message.

- In the Procedure Editor, press the **HELP** key or **F1**.
On UNIX platforms, use the OpenEdge pro command to start a single-user mode character OpenEdge client session and view a brief description of a message by providing its number.

**To use the pro command to obtain a message description by message number:**

1. Start the Procedure Editor:

   ```
   OpenEdge-install-dir/bin/pro
   ```

2. Press F3 to access the menu bar, then choose **Help > Messages**.

3. Type the message number and press **ENTER**. Details about that message number appear.

4. Press F4 to close the message, press F3 to access the Procedure Editor menu, and choose **File > Exit**.
Overview of PAS for OpenEdge administration

This section is an introduction to the overall administrative framework of PAS for OpenEdge.

Note: For additional PAS for OpenEdge documentation, including white papers on advanced topics, see: https://community.progress.com/community_groups/openedge_development/m/documents

For details, see the following topics:

• Development servers compared to production servers
• Managing the ROOT Web application
• Transport URLs
• Tomcat logging
• Configuration and properties files
• Progress Application Server properties
• Progress Application Server directories
Development servers compared to production servers

Progress Application Server for OpenEdge can be installed as either a server for developing and testing Web applications or as a production server for application deployment.

The difference between a development server and a production server is largely a matter of security. Typically, a server in a development environment allows unrestricted access to and control by a user or group of users. A production server, on the other hand, is configured to restrict access to authorized users and to limit control to system administrators. For example, in a development environment anyone might be allowed to shut down the server, whereas, in a production environment, only an administrator with appropriate privileges would be allowed to stop a running server.

In addition, you should be aware that the development server includes a number of Tomcat and Progress configuration and management tools. A production server, to avoid unauthorized tampering, would not include those tools.

See also

Development server and production server security issues on page 79

Managing the ROOT Web application

Tomcat requires a default Web application in /webapps/ROOT. In PAS for OpenEdge, the default root application is a deployment of oeabl.war. To suit the requirements of your particular development or deployment environment, you can replace ROOT with your own Web application and redeploy oeabl.war. For example, you can:

1. Delete the existing root application in a PAS for OpenEdge instance's /webapps folder.
2. Rename your Web application as ROOT.war.
3. Deploy your ROOT.war to the PAS instance.
4. Redeploy the default oeabl.war application, which can be found in $CATALINA_HOME/extras, under a folder other than ROOT.

Transport URLs

In PAS for OpenEdge, transports handle HTTP requests and are built into the default ROOT (oeabl.war) application. (A PAS for OpenEdge instance contains one or more OEABL Web applications.) There are four transports, APSV, REST, SOAP, and WEB. These transports handle requests from clients that use any one of those protocols for communication with a Web server.
The four transports and their assigned URLs shown in the following table.

**Table 1: Transport URLs**

<table>
<thead>
<tr>
<th>Transport</th>
<th>Description</th>
<th>URL Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>APSV</td>
<td>Handles requests from clients that are using the AppServer over HTTP protocol.</td>
<td>/apsv</td>
</tr>
<tr>
<td>REST</td>
<td>Handles REST RPC over CFX and Came.l</td>
<td>/rest</td>
</tr>
<tr>
<td>SOAP</td>
<td>Handles SOAP 1.1 requests.</td>
<td>/soap</td>
</tr>
<tr>
<td>WEB</td>
<td>Handles requests that use standard HTTP verbs, which includes classic WebSpeed and OpenHTTP.</td>
<td>/web</td>
</tr>
</tbody>
</table>

**Note:** If you are referencing an application other than the ROOT application, the URL path is preceded by the application name (**app_name/web**, for example).

**See also**

[Modifying the WEB transport URL on page 21](#)

**Modifying the WEB transport URL**

The WEB transport handles requests and responses that use standard HTTP verbs. This includes interaction with clients such as WebSpeed and OpenHTTP.

If you want to change the default /web URL, you can add additional handlers and map them to different URLs, or you can change the mappings of the default handlers.

The WEB transport, which supports streaming HTTP applications, supports two default URL handlers:

- The **OpenEdge.Web.CompatibilityHandler** provides compatibility with WebSpeed SpeedScript and CGI Wrapper applications. This is the default handler used in an instance in a development environment.
- The **OpenEdge.Web.DefaultWebHandler** handler returns 405 Method Not Allowed errors for requests that are not mapped to the handler. This is the default handler used in an instance in a production environment.

You can add additional handlers in the instance's **openedge.properties** file and map them to specific URLs. For example:

```plaintext
defaultHandler=OpenEdge.Web.CompatibilityHandler
handler1=SportsHandler:/mycustomer
handler2=Sportshandler:/mycustomer/{custid}
```

In addition, you can change the URL mappings of the default handlers by editing the configuration files described in the following sections.
WEB-INF/web.xml

The WEB transport is defined as a servlet in the OEABL application’s WEB-INF/web.xml file. Look for the section headed by <servlet-name>OEWebServlet</servlet-name>.

After the servlet definition, you will see a <context-param> and a <servlet-mapping> section, where you can change the default WEB URL. The example below shows in bold where the WEB URL is defined.

```xml
<!-- The context parameter and URL mapping must be consistent if you are looking to modify the URL space -->
<context-param>
    <param-name>OEWebURL</param-name>
    <param-value>/web</param-value>
</context-param>
<servlet-mapping>
    <servlet-name>OEWebServlet</servlet-name>
    <url-pattern>/web/*</url-pattern>
</servlet-mapping>
```

If you modify the default URL, make sure that both param-value (referenced by the servlet) and url-pattern (referenced by the Tomcat server) have the same URL path.

WEB-INF/oeablsecurity-*.xml

If you changed the WEB URL path in the web.xml file, you must also change it in the security configuration file that you are using.

If you are using anonymous security (oeablSecurity-anonymous.xml), then you need to change modify the default intercept URL definition that is shown below.

```xml
<intercept-url pattern="/web/**"
    access="hasAnyRole('ROLE_ANONYMOUS')"/>
```

If you are using one of the other security files, you must change the intercept URL shown above and the HTTP definition shown below.

```xml
<http pattern="/web/**"
    auto-config="true"
    use-expressions="true"
    create-session="stateless"
    disable-url-rewriting="true"
    authentication-manager-ref="oeablApplicationAuth"
    realm="OpenEdge" />
```

Note that the URL patterns specified in the security files must match the URLs specified in the web.xml file.
Tomcat logging

The core Progress Application Server (PAS) uses the standard logging technology employed by the Apache Tomcat Web server. For system logging, Tomcat uses an implementation of the Apache Commons Logging library based on the java.util.logging (JULI) framework. JULI is implemented in $CATALINA_HOME/bin/tomcat-juli.jar.

Note: This topic only applies to the standard JULI-based logging implemented for the core Tomcat Web server. Web application logging may be based on a different framework (Log4j for example).

Log files are written to the /logs directory of a running instance ($CATALINA_BASE/logs). The default log files for PAS are:

- catalina.log, a log with entries that describe server activity
- catalina.out, a log for system output and standard error messages
- localhost.log, a log file for tracking Web application activity
- localhost_access.log, a log for tracking requests processed by the server
- host_manager.log, a log for Tomcat's host-manager.war Web application
- manager.log, a log for Tomcat's manager.war Web application

Log files, with the exception of catalina.out, are saved daily with the date appended to the filename (MM-DD_YYYY). catalina.out persists while the server is running. You can delete or archive all of the log files with the TCMAN clean action.

The default log files and logging levels are defined in /conf/logging.properties. For example, the following entry from logging.properties instantiates the catalina.log file:

```
1catalina.org.apache.juli.FileHandler.level = INFO
1catalina.org.apache.juli.FileHandler.directory = ${catalina.base}/logs
1catalina.org.apache.juli.FileHandler.prefix = catalina.
```

Notice that the logging level is set to INFO which is the default for all log files in PAS. The following is a list of the JULI logging levels that apply to Tomcat.
Table 2: JULI logging levels for Tomcat

<table>
<thead>
<tr>
<th>Level</th>
<th>Log content</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEVERE</td>
<td>Serious failures</td>
</tr>
<tr>
<td>WARNING</td>
<td>Potential problems</td>
</tr>
<tr>
<td>INFO</td>
<td>Informational messages</td>
</tr>
<tr>
<td>CONFIG</td>
<td>Static configuration messages</td>
</tr>
<tr>
<td>FINE</td>
<td>Trace messages</td>
</tr>
<tr>
<td>FINER</td>
<td>Detailed trace messages</td>
</tr>
<tr>
<td>FINEST</td>
<td>Highly detailed trace messages</td>
</tr>
</tbody>
</table>

The list is arranged in order by level of detail from the least (SEVERE) to the greatest (FINEST). Note that a level includes all the content of previous levels. For example, INFO specifies that the log should include informational messages, potential problems (WARNING), and serious failures (SEVERE).

Configuration and properties files

The following table lists the configuration files that are located in the Progress Application Server’s /conf directory. Where noted, standard Tomcat configuration files were extended to support configuration of the Progress Application Server.

**Note:** Although these files can be edited in a text editor, it is always safer to use a utility like tcman.sh config to set, add, or remove properties.

Table 3: Progress Application Server properties files

<table>
<thead>
<tr>
<th>Properties</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>appserver.properties</td>
<td>An added Progress properties file that sets custom Progress Application Server properties. It is called on startup by conf/server.xml and is also used by Progress web applications. For more information, open the conf/appserver.properties.README file.</td>
</tr>
<tr>
<td>catalina.properties</td>
<td>A standard Tomcat properties file that was extended to access common shared libraries and to supply external Java properties used by server.xml. For more information, open the conf/catalina.properties.README file.</td>
</tr>
</tbody>
</table>
### Progress Application Server properties

The following tables list the configuration properties for a Progress Application Server. Except where noted, use the TCMAN utility to set, add, or remove properties.

**Note:** Where indicated in the table, more information can be found in the Tomcat configuration reference at [http://tomcat.apache.org/tomcat-7.0-doc/config/](http://tomcat.apache.org/tomcat-7.0-doc/config/).

#### Table 4: PAS properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>catalina.base</code></td>
<td>A read-only string containing the path to the PAS instance’s root directory. This property cannot be configured using TCMAN.</td>
</tr>
<tr>
<td><code>catalina.home</code></td>
<td>A read-only string containing the path to the PAS home server's root directory. This property cannot be configured using TCMAN.</td>
</tr>
<tr>
<td><code>com.sun.management.jmxremote.access.file</code></td>
<td>A readable/writable string containing the absolute path to where the PAS instance JMX console access control file is stored.</td>
</tr>
<tr>
<td><code>com.sun.management.jmxremote.password.file</code></td>
<td>A readable/writable string containing the absolute path to where the PAS instance JMX console password is stored.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>com.sun.management.jmxremote.ssl</td>
<td>A readable/writable Boolean indicating whether SSL is enabled for JMX console connections. The possible values are true (use SSL) and false (no SSL).</td>
</tr>
<tr>
<td>java.io.tmpdir</td>
<td>A read-only string containing the path to the PAS home server's temp directory, where transitory files are created.</td>
</tr>
<tr>
<td>psc.as.ajp13.maxconnections</td>
<td>A readable/writable numeric value containing the maximum number of client connections for the AJP13 connector. When the maximum number is reached, new connections are queued. The minimum value is 100. For more information, see &quot;Connector&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.ajp13.maxqueuesize</td>
<td>A readable/writable numeric value containing the maximum number of incoming requests in the AJP13 connector's queue. For more information, see &quot;Connector&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.ajp13.port</td>
<td>A readable/writable numeric containing the TCP port number for the AJP13 protocol used by a load balancing server. The value can range from 1 to the maximum port number. For more information, see &quot;Connector&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.ajp13.tomcatauth</td>
<td>A readable/writable Boolean indicating whether the AJP13 client's authentication feature is enabled. The possible values are true (enabled) and false (disabled). For more information, see &quot;Connector&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.alias</td>
<td>A read-only string containing the server's alias name, as used in TCMAN's -I option and with cluster and load balancing configurations. If the property contains home, then the server is a full PAS installation.</td>
</tr>
<tr>
<td>psc.as.autodeploy</td>
<td>A readable/writable Boolean indicating whether the PAS server automatically deploys web applications placed into its web application deployment directory (see property psc.as.webappdir). The possible values are true (autodeploy, less secure) and false (no autodeploy more secure). For more information, see &quot;Host&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.clust.expireOnShut</td>
<td>A readable/writable Boolean indicating whether sessions expire on all nodes when one node is shut down. The possible values are true (all sessions expire) and false (do not expire). For more information, see &quot;ClusterManager&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>psc.as.clust.inactivetimeout</td>
<td>A readable/writable numeric value containing the number of seconds a Tomcat cluster session can be inactive before it is marked expired. The minimum value is 10 seconds. For more information, see &quot;ClusterManager&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.clust.manager</td>
<td>A readable/writable string indicating the Tomcat cluster manager type. The possible values are DeltaManager and BackupManager. For more information, see &quot;Cluster&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.clust.mcast.addr</td>
<td>A readable/writable string containing the Tomcat cluster multicast address, formatted as an IPv4 address. For more information, see &quot;Channel/Membership&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.clust.mcast.dropafter</td>
<td>A readable/writable numeric value containing the amount of time a cluster member can be non-responsive before timing out. The minimum value is 10 milliseconds. For more information, see &quot;Channel/Membership&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.clust.mcast.freq</td>
<td>A readable/writable numeric value containing the Tomcat cluster multicast heartbeat frequency. The minimum value is 10 milliseconds. For more information, see &quot;Channel/Membership&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.clust.mcast.port</td>
<td>A readable/writable numeric value containing the Tomcat cluster multicast port number. For more information, see &quot;Channel/Membership&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.clust.notifyListeners</td>
<td>A readable/writable Boolean indicating whether Tomcat notifies session listeners when sessions are created or expired across the node. The possible values are true (notify) and false (do not notify). For more information, see &quot;ClusterManager&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.clust.recv.addr</td>
<td>A readable/writable string containing the Tomcat cluster multicast receiver address, formatted as an IPv4 address. For more information, see &quot;Channel/Receiver&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.clust.recv.autobind</td>
<td>A readable/writable numeric value containing the number of times Tomcat will increment upward if the port in psc.as.clust.recv.port is unavailable. For more information, see &quot;Channel/Receiver&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td><code>psc.as.clust.recv.maxthreads</code></td>
<td>A readable/writable numeric value containing the maximum number of threads in the Tomcat cluster message receiver (NioReceiver). The possible values range from 10 to 10,000. For more information, see &quot;Channel/Receiver&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td><code>psc.as.clust.recv.nodelay</code></td>
<td>A readable/writable Boolean indicating whether the Tomcat cluster message receiver uses the TCP_NODELAY option. The possible values are <code>true</code> (option on) and <code>false</code> (option off). For more information, see &quot;Channel/Receiver&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td><code>psc.as.clust.recv.port</code></td>
<td>A readable/writable numeric value containing the Tomcat cluster multicast receiver port number. For more information, see &quot;Channel/Receiver&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td><code>psc.as.clust.recv.selectortimeout</code></td>
<td>A readable/writable numeric value containing the time in milliseconds for timeout of the Tomcat cluster message receiver (NioReceiver). The possible values range from 10 to 10,000. For more information, see &quot;Channel/Receiver&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td><code>psc.as.clust.recv.timeout</code></td>
<td>A readable/writable numeric value containing the Tomcat cluster message receiver timeout in milliseconds. The minimum value is 10. For more information, see &quot;Channel/Receiver&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td><code>psc.as.clust.sendoptions</code></td>
<td>A readable/writable bitmask for the Tomcat cluster channel send options. The possible values are synchronized_ack, 0x0004; asynchronous, 0x0008; and use ack, 0x0002. For more information, see &quot;Cluster&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td><code>psc.as.commonlib</code></td>
<td>A readable/writable string containing a classpath of directories and JAR files that the PAS server uses with the web application class loader. The value is a comma separated list of absolute paths to directories or JAR files. For example: <code>/local/server/common/*.jar,/local/server/common/x.jar</code></td>
</tr>
<tr>
<td><code>psc.as.executor.maxthreads</code></td>
<td>A readable/writable numeric value containing the maximum number of threads in the server's HTTP thread pool. The minimum value is 100. For more information, see &quot;Executor&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>psc.as.executor.minsparethreads</td>
<td>A readable/writable numeric value containing the minimum number of threads in the HTTP/HTTPS connector's thread pool. The range of values is from 10 to <code>psc.as.executor.maxthreads</code>. For more information, see &quot;Executor&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.http.connectiontimeout</td>
<td>A readable/writable numeric value containing the maximum number of milliseconds the PAS will wait from when the client HTTP connection is made until the URI is received. The minimum value is 1000.</td>
</tr>
<tr>
<td>psc.as.http.maxconnections</td>
<td>A readable/writable numeric value containing the maximum number of client connections for the HTTP connector. When the maximum number is reached, new connections are queued. The minimum value is 100. For more information, see &quot;Connector&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.http.maxqueuesize</td>
<td>A readable/writable numeric value containing the maximum number of incoming requests in the HTTP connector's queue. For more information, see &quot;Connector&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.http.port</td>
<td>A readable/writable numeric containing the TCP port number for the HTTP protocol. For more information, see &quot;Connector&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.https.clientauth</td>
<td>A readable/writable Boolean indicating whether the SSL (HTTPS) client's authentication feature is enabled. The possible values are <code>true</code> (enabled) and <code>false</code> (disabled). For more information, see &quot;Connector&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.https.connectiontimeout</td>
<td>A readable/writable numeric value containing the maximum number of milliseconds the PAS will wait from when the client HTTPS connection is made until the URI is received. The minimum value is 1000.</td>
</tr>
<tr>
<td>psc.as.https.keyalias</td>
<td>A readable/writable string containing the alias to the server's private key stored in the SSL's keystore. For more information, see &quot;Connector&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.https.keypass</td>
<td>A readable/writable string containing the password to the server's private key stored in the SSL's keystore. For more information, see &quot;Connector&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>psc.as.https.maxconnections</td>
<td>A readable/writable numeric value containing the maximum number of client connections for the HTTPS connector. When the maximum number is reached, new connections are queued. The minimum value is 100. For more information, see “Connector” in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.https.maxqueuesize</td>
<td>A readable/writable numeric value containing the maximum number of incoming requests in the HTTPS connector's queue. For more information, see &quot;Connector&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.https.maxthreads</td>
<td>A readable/writable numeric value containing the maximum number of threads in the server's HTTPS thread pool. The minimum value is 100. For more information, see &quot;Connector&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.https.port</td>
<td>A readable/writable numeric value containing the TCP port number for the HTTPS protocol. For more information, see &quot;Connector&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.https.protocol</td>
<td>A readable/writable string indicating the default HTTPS protocol. The possible values are SSLv3 and TLS. For more information, see &quot;Connector&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.https.storeType</td>
<td>A readable/writable string indicating the type of keystore used for storing SSL private keys and certificates. The possible values are PKCS12 and JKS. For more information, see &quot;Connector&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.msg.maxpipeline</td>
<td>A readable/writable numeric value containing the maximum number of pipelined HTTP keepAlive requests before the socket is closed. A value of -1 disables the limit. For more information, see &quot;Connector&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.msg.maxpostsize</td>
<td>A readable/writable numeric value containing the maximum size in bytes of a connector's POST message body. A value of 0 disables the limit. For more information, see &quot;Connector&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.msg.timeout</td>
<td>A readable/writable numeric value containing a connector's asynchronous timeout in milliseconds. The minimum value is 10. For more information, see &quot;Connector&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>psc.as.os</td>
<td>A read-only string indicating the system PAS was installed on. The possible values are <strong>unix</strong> and <strong>windows</strong>.</td>
</tr>
<tr>
<td>psc.as.parent</td>
<td>A read-only string containing the path to the home server. If <code>psc.as.alias</code> is <code>home</code>, then <code>psc.as.parent</code> is blank.</td>
</tr>
<tr>
<td>psc.as.shut.port</td>
<td>A readable/writable numeric containing the TCP port number for shutting down the server via a TCP network client. A value of -1 is only supported on UNIX for disabling shutdown using a network client (which is more secure). Windows requires a port number greater than or equal to 1. For more information, see &quot;Server&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.shut.pwd</td>
<td>A readable/writable string containing the shared secret required by the PAS server before it will accept a shutdown command from a network client. The secret can be any non-blank ASCII string, and it is required in Windows. For more information, see &quot;Server&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td></td>
<td><strong>Caution:</strong> The value of this property is stored as cleartext.</td>
</tr>
<tr>
<td>psc.as.stuckthreadthreshold</td>
<td>A readable/writable numeric value containing the maximum number of seconds to wait before a thread is considered stuck. The minimum value is 0 (off). For more information, see &quot;Stuck Thread Detection Valve&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.type</td>
<td>A readable/writable string indicating the type of server. <strong>standard</strong> indicates a full PAS installation, and <strong>instance</strong> indicates a PAS instance.</td>
</tr>
<tr>
<td>psc.as.unpackwars</td>
<td>A readable/writable Boolean indicating whether the PAS server automatically unpacks WAR archives placed in its web application deployment directory (see property <code>psc.as.webappdir</code>). The possible values are <strong>true</strong> (automatic unpacking) and <strong>false</strong> (no automatic unpacking). For more information, see &quot;Host&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.usersfile</td>
<td>A readable/writable string containing the path where the PAS server finds the local user account storage file. The path can be absolute or relative to the server (instance) <code>CATALINA_BASE</code> configuration value. For more information, see &quot;Resources&quot; in the Apache Tomcat documentation.</td>
</tr>
</tbody>
</table>
The properties in the following table control the enabling and disabling of PAS features. All of them are settable using TCMAN, and the possible values are on and off.

**Table 5: PAS features**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccessLog</td>
<td>Indicates whether client HTTP message and response code logging is enabled.</td>
</tr>
<tr>
<td>AJP13</td>
<td>Indicates whether the AJP13 protocol network port is enabled on the server for load balancing by a Apache HTTPD or IIS web server.</td>
</tr>
<tr>
<td>Cluster</td>
<td>Indicates whether Tomcat cluster support (which needs to be manually configured) is enabled.</td>
</tr>
<tr>
<td>CrawlerSessionManager</td>
<td>Indicates whether reduction of the impact of web crawler session creation (so as not to overrun the server) is enabled.</td>
</tr>
<tr>
<td>HTTP</td>
<td>Indicates whether the HTTP protocol network port is enabled on the server.</td>
</tr>
<tr>
<td>HTTPS</td>
<td>Indicates whether the HTTPS protocol network port is enabled on the server.</td>
</tr>
<tr>
<td>JAASRealm</td>
<td>Indicates whether the use of JAAS plug-in authentication modules for container user authentication (which requires manual configuration) is enabled.</td>
</tr>
<tr>
<td>JMXLifecycle</td>
<td>Indicates whether network ports used for remote JMX console are fixed. Otherwise, standard RMI connections are used.</td>
</tr>
<tr>
<td>Apache Tomcat Directory Structure Attributes</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>LDAPRealm</td>
<td>Indicates whether the use of an LDAP/Active-Directory service for container user authentication (which requires manual configuration) is enabled.</td>
</tr>
<tr>
<td>RemoteAddrValve</td>
<td>Indicates whether connections are restricted to a list of specific TCP/IP addresses.</td>
</tr>
<tr>
<td>RemoteHostValve</td>
<td>Indicates whether connections are restricted to a list of specific DNS host names.</td>
</tr>
<tr>
<td>SecurityListener</td>
<td>Indicates whether security checking at server startup is enabled, which prevents the server from starting if any of the checks fail.</td>
</tr>
<tr>
<td>SingleSignOn</td>
<td>Indicates whether Tomcat's single-sign on feature is enabled.</td>
</tr>
<tr>
<td>StuckSessionValve</td>
<td>Indicates whether messages will be logged about requests that do not appear to be completing.</td>
</tr>
<tr>
<td>UserDatabase</td>
<td>Indicates whether the use of a flat file for container user authentication is enabled.</td>
</tr>
</tbody>
</table>

## Progress Application Server directories

The following table lists the PAS directories added to the standard Tomcat directory structure.

### Table 6: Progress Application Server directory structure extensions

<table>
<thead>
<tr>
<th>Directory Structure</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>

---

Progress Application Server for OpenEdge: Administration Guide
There are a number of tools and utilities that you can use to configure and administer PAS for OpenEdge.

For details, see the following topics:

- Tomcat utilities
- TCMAN
- PASOESTART
- PASMAN
- OEPROP
- Utilities for managing keys and digital certificates
- OpenEdge Management and OpenEdge Explorer
- JConsole and JMX

Tomcat utilities

PAS for OpenEdge includes the standard Apache Tomcat administrative utilities (startup, shutdown, configtest and so on).

However, PAS for OpenEdge includes the TCMAN command line utility as an interface to administrative functionality provided by core Tomcat scripts and more. You will find extended administrative functionality that is easier to find and to use if you run TCMAN rather than running the Tomcat scripts directly. In addition, the TCMAN administrative functionality is tailored specifically for use with PAS for OpenEdge.
TCMAN

TCMAN is a command-line utility for administering and managing the Progress application server. It extends the functionality that is provided by the core Apache Tomcat administrative scripts found in the server's /bin folder.

Syntax

```bash
{ $CATALINA_HOME | $CATALINA_BASE }/bin/tcman{.sh | .bat} action [ general_options ]
[ action_options ]
```

Parameters

$CATALINA_HOME/$CATALINA_BASE

$CATALINA_HOME is the path of the directory where the core PAS server is installed, typically OpenEdge-Install-Dir/servers/pasoe in PAS for OpenEdge. You always run TCMAN from $CATALINA_HOME/bin when you are creating new instances or worker configuration files. However, you can add the -I instance_alias_name option to specify an instance on which to apply the TCMAN action.

$CATALINA_BASE is the path of an instance's directory.

Note:

TCMAN automatically determines the value of $CATALINA_BASE based on the directory where you start it. When you run it from the /bin directory of an instance, the value of $CATALINA_BASE is the root directory of the instance.

If you run TCMAN from the core PAS server (OpenEdge-Install-Dir/servers/pasoe/bin), the value of $CATALINA_BASE and $CATALINA_HOME are the same.

action

Specify which TCMAN action to invoke.

general_options

Specify one or more of the TCMAN common options that can apply to most actions.

The output of `tcman.sh help action` includes a list of general options that are applicable to a particular action.

See also

TCMAN on page 36
OEPROP on page 45
action_options

Specify an option that applies to the selected action. These options are explained in the topics that describe each action.

See also
TCMAN Reference on page 115

PASOESTART

PASOESTART is an OpenEdge extension to the TCMAN utility that encapsulates a number of separate startup actions into a single command.

Starting a PAS for OpenEdge instance usually involves manually performing the following individual actions:
1. Executing `tcman clean -A` to clean and archive log files
2. Running `tcman start` to launch the instance
3. Executing `tcman env` to check if the instance is running
4. Polling the core server to report that the instance's startup process has completed
5. Scanning the core server and web application context logs for severe conditions that render the server unusable
6. Verifying that at least one child OS process, in the form of an ABL multi-session Agent, is running
7. Scanning the multi-session Agent's log file for startup errors that render the Agent unusable

The PASOESTART command produces a more stable and predictable startup process by performing the TCMAN actions described in the first three steps, and then by performing the error checking and process validation described in the final four steps. It executes until the instance starts without any fatal errors, or until a timeout limit has been reached and the startup process is aborted.

Note: The core Apache Tomcat server in PAS for OpenEdge was designed to start even when severe network connection failures or invalid web application contexts render the server unusable by clients. Using PASOESTART helps to ensure that the instance is stable and usable after startup.

Output from the PASOESTART action includes all the status and error messages generated during the various steps in the startup process.

Syntax

Executing PASOESTART from PASMAN:

```
oe-install-dir/bin/pasman{.sh|.bat} pasoestart -I instance-name
[ -timeout seconds] [ -v] [ -silent] [ -json] [ -restart] [ -archive] [ -nostop]
```
Executing PASOESTART from TCMAN:

```
instance-dir/bin/tcman{.sh|.bat} pasoestart
```

**Parameters**

- `instance-name`
  
  The name of the instance to start.

  **Note:** This parameter is required when you execute PASOESTART with PASMAN. This parameter is not necessary when you execute PASOESTART with TCMAN from the instance’s `instance-name/bin` directory.

- `timeout seconds`
  
  The number of seconds to wait for the PAS for OpenEdge instance to start. If the instance has not started fully within the time-out interval, it is considered hung and will be stopped.

  If not specified, the time-out interval is set to a default of 30 seconds on UNIX servers and 120 seconds on Windows servers. If the interval is set to 0 (zero), only the current state of the specified instance is displayed. Negative values are not allowed and will generate a command-line option error.

- `v`
  
  Display verbose execution information to the console.

- `silent`
  
  Direct execution information to a log file (`instance-name/logs/pasoestart.log`). Note that status information is still sent to the console when you use `-silent`. See Status Information.

  **Note:** Status information is still sent to the console when you use `-silent`. See Status Information.

- `json`
  
  Format completion status information as a JSON object and send it to the console. The default format is multi-line TEXT output to the console.
The following shows the format of the JSON output:

```json
{
  "start-action":"startup-action",
  "initial-state":"pasoe-state",
  "initial-processes":"os-process-list",
  "exit-state":"pasoe-state",
  "exit-description":"execution-status",
  "exit-processes":"os-process-list",
  "exit-status":"exit-status",
  "exit-errors":
    [  "line-of-error-text"
    ]
}
```

-restart

If the instance is already running or is in a hung state, attempt to stop it, and then execute a full startup. If the instance is already in the stopped state, this option has no effect

-archive

Archive existing log files. This option executes the `tcman clean -A` command.

-nostop

Turns off the default stop action, which automatically occurs after a timeout, or when PASOESTART detects errors in an instance’s log files.

**Note:** -nostop can start an instance that is hung or in an unstable state. The primary purpose of this option is to force a problematic instance to start in order to debug issues. You should not use it for standard startups.

### Status Codes

PASOESTART is always a blocking operation. Control is not returned to the shell until operations are complete and the PAS for OpenEdge instance is in a known state. (See [Status Information](#) for more on known states.) After PASOESTART completes, the instance’s startup completion status is in the OS error variable (%errorlevel% for Windows, $? for UNIX).

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The instance started/restarted without errors and is usable in a running state.</td>
</tr>
<tr>
<td>1</td>
<td>The core (Tomcat) server process failed to start and the instance is left in the stopped state.</td>
</tr>
<tr>
<td>2</td>
<td>The instance started, but encountered errors that make some/all of it unusable, and is left in the stopped state for troubleshooting and problem resolution.</td>
</tr>
<tr>
<td>3</td>
<td>The instance was already in the running state, was not restarted, and was left in a running state that may, or may not, be usable.</td>
</tr>
</tbody>
</table>
The instance startup was attempted from a stopped state, but timed out before it reached a running state. The instance is left in the stopped state for troubleshooting and problem resolution.

The instance was hung in the starting state, was not restarted, and was left in the hung starting state.

The instance was hung in the stopping state, was not restarted, and was left in the hung stopping state.

Command-line line option error(s) prevented execution of PASOESTART.

PASOESTART was used to start a Windows Service and no startup process was attempted.

Missing OS process environment variables prevented PASOESTART from running and no startup process was attempted.

An unspecified internal programming error was encountered. Contact OpenEdge technical support for more information.

### Status Information

PASOESTART always sends status information to the console, even when you use the `-silent` option. A full status report includes following information fields, which are available either as human-readable text or as a JSON object:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Start action</strong></td>
<td>The type of startup process executed: `{ start</td>
</tr>
<tr>
<td><strong>Initial state</strong></td>
<td>The initial state of the PASOE instance found at execution time</td>
</tr>
<tr>
<td></td>
<td>`{ stopped</td>
</tr>
<tr>
<td><strong>Initial processes</strong></td>
<td>A space separated list of the PASOE instance's OS process ids found at execution time:</td>
</tr>
<tr>
<td></td>
<td>A single 0 value indicates no OS processes (i.e. stopped)</td>
</tr>
<tr>
<td></td>
<td>A single non-zero value indicates the PASOE server is running but no multi-session Agents (i.e. starting or stopping)</td>
</tr>
<tr>
<td></td>
<td>Multiple non-zero values indicts the PASOE server and at least one Agent is running – however errors may be present that prevent the instance for serving clients</td>
</tr>
<tr>
<td><strong>Exit state</strong></td>
<td>The PASOE instance’s state when the pasoestart command completes. See Initial state</td>
</tr>
<tr>
<td><strong>Exit description</strong></td>
<td>A text description of the command completion status</td>
</tr>
</tbody>
</table>
Exit status | A single numerical value of the pasoestart command’s completion status. See the above list of `pasoestart` exit statuses
---|---
Exit errors | A brief list of error or fatal log file entries found in the PASOE instance’s /logs files. This is provided as hints as to why the PASOE instance startup failed and provides references for where the administrator/developer may find the full error output. The Exit errors information may be blank/empty for some types of completion statuses

**Server State Transitions**

PASOESTART may transition the state of a PAS for OpenEdge instance zero or more times during its execution. The following table shows the different initial and exit state transitions an instance may go through.

<table>
<thead>
<tr>
<th>Initial State</th>
<th>Exit State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stopped</td>
<td>started</td>
<td>PASOESTART successfully started Tomcat, the OpenEdge application, the Agent, and startup event procedures</td>
</tr>
<tr>
<td>stopped</td>
<td>stopped</td>
<td>Startup failed and the instance was stopped due to a PASOESTART timeout or errors.</td>
</tr>
</tbody>
</table>
| stopped       | starting   | A startup timeout was detected and the server is hung in the starting state after a failed attempt to stop the instance.  
 Try the PASOESTART `restart` command once, then manually stop the instance if restart is not successful. |
| started       | started    | The instance was already running – the state of the instance is not changed. The server may, or may not, be usable |
| started       | started    | After a PASOESTART `restart`, a running instance was stopped, and then Tomcat, the OpenEdge application, the Agent, and startup event procedures were restarted successfully. |
| started       | stopped    | After a PASOESTART `restart`, a running instance was stopped, and failed during a restart due to a timeout or errors.  
 Try the PASOESTART `restart` command once more, then manually stop the instance if restart is not successful. |
| started       | starting   | After a PASOESTART `restart`, a running instance was stopped, and timed out in the `starting` state.  
 The instance is not in a usable condition. Retry the PASOESTART `restart` command once. If it is not successful, manually stop the instance. |
<table>
<thead>
<tr>
<th>State</th>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>starting</td>
<td>starting</td>
<td>After a PASOESTART restart, an instance is hung in the starting state and no restart was performed. The instance is not in a usable condition. If it is not successful, manually stop the instance.</td>
</tr>
<tr>
<td>starting</td>
<td>stopping</td>
<td>The PASOE instance is hung in the starting state and no restart was performed. The instance is not in a usable condition. Manually stop the instance or run a PASOESTART restart.</td>
</tr>
<tr>
<td>starting</td>
<td>stopping</td>
<td>After attempting to stop an instance hung in the starting state, the instance hangs in the stopping state. The instance is not in a usable condition. Try the PASOESTART restart command once. If it is not successful, manually stop the instance.</td>
</tr>
<tr>
<td>starting</td>
<td>stopped</td>
<td>After a PASOESTART restart, an instance hung in the starting state, was stopped, and failed to restart due to a timeout or errors. Retry the PASOESTART restart command once.</td>
</tr>
<tr>
<td>starting</td>
<td>started</td>
<td>After a PASOESTART restart, an instance hung in the starting state was stopped and then Tomcat, the OpenEdge application, the Agent, and startup event procedures were restarted successfully.</td>
</tr>
<tr>
<td>stopping</td>
<td>stopping</td>
<td>The instance is hung in the stopping state and no restart was performed. The instance is not in a usable condition. Manually stop the instance or run a PASOESTART restart.</td>
</tr>
<tr>
<td>stopping</td>
<td>stopping</td>
<td>After a PASOESTART restart, an instance hung in the stopping state, was stopped, restarted, and then hung in the stopping state because of a timeout or errors. The instance is not in a usable condition. Manually stop the instance.</td>
</tr>
<tr>
<td>stopping</td>
<td>stopped</td>
<td>After a PASOESTART restart, an instance hung in the stopping state, was stopped, restarted, and stopped a second time because of a timeout or errors. The instance is not in a usable condition.</td>
</tr>
<tr>
<td>stopping</td>
<td>started</td>
<td>After a PASOESTART restart, an instance hung in the stopping state, was stopped, and then successfully restarted and then Tomcat, the OpenEdge application, the Agent, and startup event procedures were restarted successfully.</td>
</tr>
</tbody>
</table>
PASMAN

PASMAN is a command-line utility that always invokes TCMAN from the core PAS for OpenEdge installation (OpenEdge-Install-Dir/servers/pasoe/bin). You can apply TCMAN actions to an instance by using the -I instance_name option with PASMAN.

The script for PASMAN is located in OpenEdge-Install-Dir/bin which is usually included in your PATH environment variable.

PASMAN takes the same options and actions as TCMAN. However, you must specify an instance name (using -I instance_name) whenever an action applies to a specific instance.

Syntax

```
pasman action [general_options] [action_options]
```

Parameters

action

Specify which TCMAN action to invoke. See the TCMAN Reference on page 115 for information on actions.

general_options

Specify one or more of the TCMAN common options that can apply to most actions.

The output of tcmansh help action includes a list of the general options that are applicable to the specified action.
Important:
When an action affects a specific instance, you must explicitly reference the instance using the `-I instance_name` general option with PASMAN. If you do not specify an instance name, the action applies to the core server in `$CATALINA_HOME`. The actions where you can use `-I` with PASMAN to specify a particular instance are:

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clean</td>
<td>Clean up or archive log files.</td>
</tr>
<tr>
<td>config</td>
<td>Display and manage configurations</td>
</tr>
<tr>
<td>deploy</td>
<td>Deploy a Web application</td>
</tr>
<tr>
<td>disable</td>
<td>Stop a Web application</td>
</tr>
<tr>
<td>enable</td>
<td>Start a Web application</td>
</tr>
<tr>
<td>env</td>
<td>Display runtime environment information</td>
</tr>
<tr>
<td>feature</td>
<td>Display or modify server features</td>
</tr>
<tr>
<td>info</td>
<td>Display OS and server information</td>
</tr>
<tr>
<td>leaks</td>
<td>Display memory leaks</td>
</tr>
<tr>
<td>list</td>
<td>List deployed applications</td>
</tr>
<tr>
<td>plist</td>
<td>List process ids</td>
</tr>
<tr>
<td>reload</td>
<td>Reload a Web application</td>
</tr>
<tr>
<td>resources</td>
<td>Display global server resources</td>
</tr>
<tr>
<td>sessions</td>
<td>Display Web application HTTP sessions</td>
</tr>
<tr>
<td>showproc</td>
<td>Show Windows process information</td>
</tr>
<tr>
<td>start</td>
<td>Start an instance</td>
</tr>
<tr>
<td>status</td>
<td>Display detailed server status</td>
</tr>
<tr>
<td>stop</td>
<td>Stop an instance</td>
</tr>
<tr>
<td>test</td>
<td>Test a server configuration</td>
</tr>
<tr>
<td>undeploy</td>
<td>Undeploy a Web application</td>
</tr>
<tr>
<td>version</td>
<td>Display server, OS, and runtime version information</td>
</tr>
</tbody>
</table>
Other actions only apply to the core PAS for OpenEdge installed in
`OpenEdge-Install-Dir/servers/pasoe/bin` and not to instances. These core server actions,
where `-I instance_name` should not be used with PASMAN, are:

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>create</td>
<td>Create an instance</td>
</tr>
<tr>
<td>delete</td>
<td>Delete an instance</td>
</tr>
<tr>
<td>instances</td>
<td>Display server instances</td>
</tr>
<tr>
<td>register</td>
<td>Register an instance for tracking</td>
</tr>
<tr>
<td>service</td>
<td>Register and manage an instance as a Windows service</td>
</tr>
<tr>
<td>unregister</td>
<td>Stop tracking an instance</td>
</tr>
<tr>
<td>workers</td>
<td>Create a Tomcat worker configuration file</td>
</tr>
</tbody>
</table>

*action_options*

Specify an option that applies to the selected *action*. These options are explained in the topics that describe each action.

**See also**

TCMAN Reference on page 115

---

**OEPROP**

**Purpose**

OEPROP is a command-line utility used to display and manage properties in the server's
`conf/openedge.properties` file.

**Note:** Use TCMAN (tcmn config) to manage the properties in `conf/appserver.properties` or to view the properties in `conf/jvm.properties`.

**Syntax**

```
{ $CATALINA_HOME | $CATALINA_BASE }/bin/oeprop [.sh | .bat] [-help]
[group_name.prop_name | group_name.prop_name=value | +group_name.prop_name=value
| ~group_name.prop_name | ~group_name] [-R] [-f filename] [-v] [-g] [-t]
```
Parameters

- help

Display the syntax and parameter descriptions for the oeprop command.

`group_name.prop_name`

Display the value of the specified property the the specified group in ...
conf/openedge.properties.

`group_name.prop_name=value`

Set the value of a property in a group that exists in ...
conf/openedge.properties.

`+group_name.prop_name=value`

Add a new property to the specified group in ...
conf/openedge.properties and set its value.

`~group_name.prop_name`

Remove the specified property in the specified group in ...
conf/openedge.properties. If prop_name refers to a group, the group must be empty. To remove non-empty groups, use the -R option.

`~group_name`

Remove the specified group in ...
conf/openedge.properties. The group must be empty. To remove non-empty groups, add the -R option.

-R

Use with ~group_name to remove non-empty groups.

-f filename

Merge the contents of the text file, filename, with the existing openedge.properties file. This option is particularly useful when deploying and configuring new Web applications. It allows you to add property names and property groups in one step.

-v

Display verbose information when oeprop runs.

-g

Display debug information when oeprop runs.

-t

Display detailed trace information when oeprop runs.
Note
Properties in the openedge.properties files are organized in groups. For example, the following shows the properties listed under the AppServer.SessMgr group:

```
[AppServer.SessMgr]
  agentExecFile=${psc.as.oe.dlc}/bin/_mproapsv
  agentListenerTimeout=300000
  agentLogEntryTypes=
  agentLogFile=
  agentLoggingLevel=2
  agentLogThreshold=0
  agentNumLogFiles=3
  agentStartupParam=-T ${catalina.base}/temp
  connectionWaitTimeout=3000
  idleAgentTimeout=300000
  idleConnectionTimeout=300000
  idleResourceTimeout=0
  idleSessionTimeout=300000
  ipver=IPv4
  maxABLSessionsPerAgent=200
  maxAgents=10
  maxConnectionsPerAgent=16
  numInitialAgents=1
  publishDir=${catalina.base}/openedge
  requestWaitTimeout=15000
  socketTimeout=3000
```

When you specify a property as a parameter to the oeprop command, the group name must precede the property name. For example: AppServer.SessMgr.maxAgents.

Examples

- Display the maximum number of agents (returns 10, the default setting):

  proenv> bin\oeprop.bat AppServer.SessMgr.maxAgents
  10

- Set the maximum number of agents to 12 and verify the change:

  proenv> bin\oeprop.bat AppServer.SessMgr.maxAgents=12
  proenv> bin\oeprop.bat AppServer.SessMgr.maxAgents
  12

- Add the groups and properties contained in NewWebAppProperties.txt to conf/openedge.properties:

  proenv> bin\oeprop.bat -f NewWebAppProperties.txt
Utilities for managing keys and digital certificates

The following OpenEdge utilities, typically run from the PROENV command line, allow you to install and manage keys and digital certificates:

• **certutil** — Provides all the functions necessary to install and manage root certificates from any Certification Authority (CA) as entries in the root certificate store of an OpenEdge client machine (located in `OpenEdge-Install-Dir\certs`).

• **gendomreg** — Creates an encrypted domain registry file that can be used to seal a client-principal token. It is most often used when the token must be validated against multiple domains.

• **genpassword** — Accepts the clear-text value of a password and generates the encoded and encrypted form for the specified password.

**Note:** Also see the **stspwdutil** utility in the *Getting Started: OpenEdge Authentication Server Guide*, *stspwdutil* features similar functionality but better encryption for the PAS for OpenEdge and OpenEdge Authentication Gateway servers.

• **mkhashfile** — Provides a simple way to install a root certificate in the OpenEdge root certificate store of a client machine. Such a certificate can be authorized by your own internal-use Certification Authority (CA) or by any CA that can provide you with a PEM-encoded certificate.

• **pkiutil** — Provides all of the functions necessary to create and manage key store entries for OpenEdge SSL servers. It creates these entries from pairs of private keys and digital certificates that it stores in the OpenEdge server key store (located in `OpenEdge-Install-Dir\keys`).

You can display usage information for any of these utilities by running them with the **-help** option. For more detailed information, see the *Command and Utility Reference* in *OpenEdge Getting Started: Installation and Configuration*.

**See also**

*Obtaining a certificate from a Certificate Authority* on page 82

OpenEdge Management and OpenEdge Explorer

Much of the functionality supplied by command-line utilities like TCMAN and OEPAS is also implemented in the OpenEdge Management (OEM) and OpenEdge Explorer (OEE) browser-based tools. You can also use OEM/OEE to monitor PAS server performance and display performance statistics.

For more information, see *Configuring and Administering PAS for OpenEdge with OEE/OEM*. 
JConsole and JMX

JConsole (the Java Monitoring & Management Console), which is an application included with Java, can be used to monitor and manage PAS for OpenEdge instances. JConsole is usually used on development servers for debugging. On production servers, it is commonly used for monitoring and tuning, and for removing hung sessions.

You should not use JConsole on a local production server because it uses significant resources. Using JConsole with a remote connection to a production server avoids this problem. However, enabling remote connections requires more attention to security since it opens the door to unauthorized access to the server.

JMX

The Progress Application Server supports the use of The Java Management Extensions (JMX) technology, which facilitates dynamic access to applications and other resources. This access is accomplished through the use of Java objects called Managed Beans, or MBeans. One or more MBeans instrument a resource through the use of the MBean's attributes, actions (defined as methods), and notifications.

The instrumented resources are listed in the MBeans tab of the Java Monitoring & Management Console. For example:

For more detailed information on JMX and MBeans, see the Java documentation at http://docs.oracle.com/javase/8/docs/technotes/guides/jmx/index.html.

See also
Running JConsole remotely on page 50
Running JConsole locally

Typically, you would run JConsole locally to monitor applications running on a PAS for OpenEdge instance in a development environment. Running JConsole locally to monitor deployed applications on a production server consumes too many resources and would adversely affect performance.

1. Start PROENV and enter `jconsole` on the command line.

   The executable, `jconsole.exe` resides in the `OpenEdge-Install-dir\jdk\bin` folder, which should be in your PROPATH.

   The JConsole New Connect dialog opens, which should look similar to the following:

   ![JConsole New Connect dialog](image)

   - **Local Process**
     - `com.amazon.adrive.client.localsvc.AWingLocalService --pipeName...` with PID 9196
     - `org.apache.catalina.startup.Bootstrap PAS for OpenEdge start` with PID 12812
     - `sun.tools.jconsole.JConsole` with PID 5608

2. Select **Local Process**.

3. Select the PAS for OpenEdge process from the list of Java processes.

   The Java Monitoring & Management Console appears.

Running JConsole remotely

One of the reasons for running JConsole remotely is to minimize the effect of JConsole on server performance, since JConsole does consume a lot of system resources. However, you should be aware of the security impact of running JConsole remotely and take appropriate measures to guard against unauthorized access.

This procedure tells you how to enable JMX remote access to a PAS for OpenEdge instance, which is a prerequisite for running the JConsole administrative tool remotely.

1. Turn on the JMXLifecycle feature in a PAS for OpenEdge instance's `/conf/server.xml` file.

   The best way to turn on the feature is to use the instance's TCMAN utility. For example:

   ```bash
   $CATALINA_BASE/bin/tcman feature JMXLifecycle=on
   ```
TCMAN enables the JMX remote process and also opens ports for remote connection (JNDI 10001 and RMI 10002). You can change these ports by manually editing the JMXLifecycle section in server.xml.

2. Add the name of the instance's host to its /conf/appserver.properties file. Add and set the following property in appserver.properties:

   ```
   java.rmi.server.hostname=hostname
   ```

   **Note:** If you do not set `java.rmi.server.hostname`, the default `localhost` will be used.

3. Restrict read/write access to the instance's /conf/jmxremote.password file to the user with owner privileges.

   **Important:** `jmxremote.password` is a plain-text file that contains the username/password combination that you need in order to log in to the instance from a remote JConsole. The default username and password (`jmxadmin`, `jmxadmin`) should be changed for security reasons.

   On Windows, you can change permissions on the **Security** tab of the file's **Properties** dialog. On a UNIX system, you would run the following command:

   ```
   chmod 600 jmxremote.password
   ```

4. On a remote machine, browse to `$JAVA_HOME/bin` and run `jconsole.exe`. The JConsole New Connection dialog appears.
a) Click the Remote Process button.
b) Specify hostname and port of the instance that you want to manage.
c) Enter the credentials that are specified in the instance's jmxremote.password file.

Upon successful login, the Java Monitoring & Management Console appears.
Working with 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 OpenEdge uninstall.

For details, see the following topics:

- Overview of instances in PAS for OpenEdge
- Instance management with TCMAN
- Creating instances with TCMAN
- Installing and running an instance as a UNIX daemon
- Installing and running an instance as a Windows service
Overview of instances in PAS for OpenEdge

After you install the core Progress Application Server, you can create an instance.

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

The following figure illustrates the creation of multiple instances using the TCMAN command-line utility (with syntax simplified).

Figure 1: Generating PAS instances

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.

There are a number of advantages when 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 when you uninstall a Progress PAS product.

Some additional 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.

• You can package an instance as a Web application and deploy it to other PAS core servers.
You use `$CATALINA_HOME/bin/tcman.sh create` command to create a new instance.

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`

However, each instance is created with:

- A `$CATALINA_BASE/bin/` directory with its own copy of some of the scripts from the core PAS. These include scripts for start up, shut down, deployment, running TCMAN actions, and so on.
- A `$CATALINA_BASE/conf/` directory with its own copy of properties and configuration files.
- A `$CATALINA_BASE/webapps/` which initially only contains the `ROOT` Web application.
- A number of directories that are initially empty. These include `/logs`, `/temp`, `/work`, and `/common/lib`.

---

**Instance management with TCMAN**

TCMAN includes actions for configuring, starting, stopping, monitoring, and deleting instances.

The following table is a brief description of the instance management actions that you can perform with TCMAN. Entries link to the reference topics that provide more details, syntax, and examples.

<table>
<thead>
<tr>
<th>Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>create</td>
<td>Create an instance of the Progress Application Server.</td>
</tr>
<tr>
<td>delete</td>
<td>Remove the directory tree and all of the files in an instance.</td>
</tr>
<tr>
<td>start</td>
<td>Start an instance of a Progress Application Server.</td>
</tr>
<tr>
<td>stop</td>
<td>Stop a running instance.</td>
</tr>
<tr>
<td>config</td>
<td>View, add, update, or delete the property values specified in <code>../conf/appserver.properties</code>.</td>
</tr>
<tr>
<td>test</td>
<td>Displays information on the configuration and environment of an instance. It also displays information about error conditions.</td>
</tr>
<tr>
<td>instances</td>
<td>Display all the instances created from the Progress Application Server installed in <code>$CATALINA_HOME</code>.</td>
</tr>
<tr>
<td>unregister</td>
<td>Stop tracking an instance by removing the instance's entry from the `$CATALINA_HOME/conf/instances.[unix</td>
</tr>
</tbody>
</table>
Creating instances with TCMAN

Before you can create an instance of the Progress Application Server (PAS) using the TCMAN command line utility, you must:

- Install the core Progress Application Server
- Install JDK or JRE version 1.7 or later
- Set the `JAVA_HOME` or `JRE_HOME` environment variable to the JDK/JRE install directory.

**Note:** TCMAN is a Progress extension of the basic Tomcat administrative utilities. TCMAN simplifies instance creation and management.

An instance runs the Tomcat executable of a core PAS, but it runs in a separate JVM, is configured with its own unique ports, and other properties. (You should not attempt to use the installed, core PAS as a development or production server.) Instances allow you to run a variety of server configurations without corrupting the files in the core server. They also allow you to update the core server without re-deploying or re-configuring your Web applications.

To create an instance using the TCMAN utility:

1. Open a command shell and navigate to `$CATALINA_HOME/bin`.
   - `$CATALINA_HOME` is the directory where you installed the core Progress Application Server.
2. Run `tcman.sh create basepath` (or `tcman.bat` on Windows systems).
   - The `base_path` parameter specifies the path name where you will create the instance. It is the only required parameter for the `create` action. If you are creating multiple running instances, you should override the default port assignments by specifying the following parameters:

<table>
<thead>
<tr>
<th>Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>register</td>
<td>Register an instance for tracking purposes. (Note that instances are registered for tracking by default when they are created. The register action is only necessary if you explicitly unregistered an instance.)</td>
</tr>
<tr>
<td>clean</td>
<td>Truncate, move, or delete the log files located in the <code>/logs</code> directory of either the core server or an instance.</td>
</tr>
<tr>
<td>version</td>
<td>Show the Apache Tomcat runtime version and OS information for an instance.</td>
</tr>
</tbody>
</table>

You can also activate these ports:

- `-s port_num` Specify the TCP port to use to stop an instance. (Required on Windows systems, optional on UNIX)
Specify the TCP port that listens for AJP13 messages, an Apache protocol for handling requests from a web server to an application server. (Optional on both Windows and UNIX systems)

See Create an instance (create) on page 132 for information about other parameters.

3. (Optional) Deploy remote management applications from $CATALINA_HOME/extras to the instance.

Remote management applications are not pre-installed, and installing them is a security decision. For example, you might want to eliminate access to the configuration and control of instances by not deploying management applications to production servers, while deploying management applications to development servers.

To deploy a management application:

a) Open a command shell and navigate to $CATALINA_BASE/bin.

b) Run tcman.sh deploy '$CATALINA_HOME/extras/admin_webapp.war'.

The admin_webapp.war can be one of the following:

<table>
<thead>
<tr>
<th>host-manager.war</th>
<th>A Tomcat administration application used to get server information and provide other functionality. It should not be necessary to deploy host-manager.war if you are using the TCMAN utilities.</th>
</tr>
</thead>
<tbody>
<tr>
<td>manager.war</td>
<td>A Tomcat administration application which you must deploy in order to run some TCMAN actions. See the TCMAN Reference material for information on which TCMAN actions require deployment of manager.war.</td>
</tr>
<tr>
<td>Progress applications</td>
<td>Progress products can have web applications that enable the use of their own administrative tools.</td>
</tr>
</tbody>
</table>

For example the following command line creates an instance of /psc/pashome in /psc/acmel and specifies its ports:

```
$: /psc/pashome/bin/tcman.sh create -p 8501 -P 8601 -s 8701 /psc/acmel
Server instance acmel created at /psc/acmel
```

See also

Server actions on page 132
Create an instance (create) on page 132
Delete an instance (delete) on page 134
Display and manage an instance’s configuration (config) on page 135
Display or modify the server features of an instance (feature) on page 137
Clean up or archive server log files (clean) on page 139
Display server instances (instances) on page 140
Register an instance for tracking (register) on page 142
Installing and running an instance as a UNIX daemon

A daemon process (called a service on Windows systems) is an application without a user interface that runs in the background and responds to requests. Web servers like PAS and Tomcat typically run as Windows services or UNIX daemons.

Note: If you run a PAS instance with `tcman.sh start`, the instance runs in the context of the command shell process. It is not available as a system service that can handle external client requests. The instance must be installed as a daemon process before you can run it as a functioning Web server.

The file `$CATALINA_HOME/bin/daemon.sh` can be used as a template for starting Tomcat automatically at boot time as a child of the `init` process. For more information, see: https://tomcat.apache.org/tomcat-7.0-doc/setup.html#Unix_daemon

However, you will need to consult with a system administrator before you can configure and run PAS as a daemon process due to differences among UNIX systems and because you need administrative privileges for access to the system.

Installing and running an instance as a Windows service

To install a Progress Application Server (PAS) instance as a Windows service, you must have administrator privileges. On systems with User Account Control (UAC), you must disable UAC as well.

A service (called a daemon process on UNIX systems) is an application without a user interface that runs in the background and provides core operating system functionality. Web servers like PAS and Tomcat typically run as Windows services or UNIX daemons.

Note: If you run a PAS instance with the TCMAN `start` action, the instance runs in the context of the command shell process. It is not available as a system service that can handle external client requests. The instance must be registered as a Window service before you can start it as a service.

This is a summary of how to register and run a PAS instance as a Windows service:

1. Open a command prompt window.
2. Navigate to the core PAS `/bin` directory (`$CATALINA_HOME/bin`).
3. Run the TCMAN `service` action specifying an instance name and the `register` parameter. For example:

   ```
tcman service oepas1 register
   ```
where `oepasl` is the name of the default instance created when you installed PAS for OpenEdge.

4. Run the TCMAN `service` action specifying an instance name and the `start` parameter. For example:

```
tcman service oepasl start
```

**Note:** You can also use the TCMAN `service` action to check the running status, stop, and unregister a PAS for OpenEdge instance as a Windows service. You can also use the Windows Microsoft Management Console (MMC) or the `sc config` command to start, stop, and check the status of a service.

**See also**

Register and manage an instance as a Windows service (service) on page 144
Application deployment and management

For details, see the following topics:

- Web application management with TCMAN
- OpenEdge Web application deployment
- OpenEdge ABL applications

Web application management with TCMAN

TCMAN includes a number of actions for deploying and managing Web applications to instances.

The following table is a brief description of the Web application management actions that you can perform with TCMAN. Entries link to the reference topics that provide more details, syntax, and examples.

<table>
<thead>
<tr>
<th>Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>list</td>
<td>Display all the Web applications that are deployed on an instance.</td>
</tr>
<tr>
<td>info</td>
<td>Display server and OS information for a running instance.</td>
</tr>
<tr>
<td>deploy</td>
<td>Deploy a Web application (.war file) to a Progress Application Server instance whether the server is running (online) or is not running (offline).</td>
</tr>
<tr>
<td>undeploy</td>
<td>Remove a Web application from running (online) or stopped (offline) instances.</td>
</tr>
</tbody>
</table>
### OpenEdge Web application deployment

The `deployREST.sh` and `deploySOAP.sh` scripts allow you to add REST or SOAP services to a deployed OpenEdge ABL Web application (oeabl.war). When you invoke either of the scripts, the OpenEdge ABL Web application must be running. After you deploy the service, you must stop and restart the Progress Application Server instance to run the new service.

### REST service deployment (deployREST.sh)

**Purpose**
Deploys REST services.

**Syntax**

```
$CATALINA_BASE/bin/deployREST.sh source_descriptor service_name
```

**Parameters**

*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.
Example

$CATALINA_BASE/bin/deployREST.sh CustomerService.paaR ROOT

SOAP service deployment (deploySOAP.sh)

Purpose
Deploy SOAP services.

Syntax

$CATALINA_BASE/bin/deploySOAP.sh source_descriptor service_name

Parameters

source_descriptor

Specify the path of the source descriptor, which is a WSM file.

service_name

Specify the target service name.

Example

$CATALINA_BASE/bin/deploySOAP.sh CustomerService.wsm ROOT

Packaging web applications

The Progress Application Server supports a number of different ways for packaging web applications. Web applications can be packaged as:

- a .war file
- a .zip file that includes the necessary directory structure (see below for more information)
- a UNIX .tar file that includes the necessary directory structure
- a deep copy of the web application's directory
- a soft link (symbolic link) to the web application's directory

The standard directory structure for a web application is:

- /exampleapp - The root directory for the application is located in the server's /webapps directory (default) or another location configured by the administrator. The root directory contains the /WEB-INF folder and
any files that have to be visible to the client browser, like HTML and JSP pages. These files can be organized into a subdirectory for more complicated applications.

- /exampleapp/WEB-INF - In addition to the directories listed below, this folder contains the Web Application Deployment Descriptor (web.xml), which contains the configuration information for the application in a standard format.

- /exampleapp/WEB-INF/classes - This folder contains any Java class files that are not organized into .jar files, including servlet classes.

- /exampleapp/WEB-INF/lib - This folder contains any .jar libraries needed for the application.

**Note:** The minimum directory structure required for a web application includes the /exampleapp/WEB-INF directory and the web.xml file. The /WEB-INF/classes and /lib directories are only required when distributing loose Java class files and .jar files.


### OpenEdge ABL applications

An ABL application in the context of a PAS for OpenEdge server is a business application having:

- A unique name which can be referenced from administrative tools
- A set of r-code files that comprise the application
- A single PROPATH value that is used to access the application's r-code
- One or more OpenEdge database connections
- A set of start-up options

One or more ABL applications can execute within the boundaries of a single PAS for OpenEdge Session Manager and the mult-session agent[s] that it controls.

**Note:** For additional PAS for OpenEdge documentation, including white papers on advanced topics, see: [https://community.progress.com/community_groups/openedge_development/m/documents](https://community.progress.com/community_groups/openedge_development/m/documents)
HTTP sessions

HTTP sessions are a standard method for maintaining context during a series of requests/responses between an HTTP client and a Web application.

For details, see the following topics:

• Overview of HTTP sessions
• Clusters and sticky sessions

Overview of HTTP sessions

HTTP sessions is an industry standard feature that allows Web servers to maintain user identity and to store user-specific data during multiple request/response interactions between a client application and a Web application. HTTP sessions preserves:

• Information about the session itself (session identifier, creation time, time last accessed, etc.)
• Contextual information about the user (client login state, for example, plus whatever else the Web application needs to save)

The Tomcat servlet container, which is built into PAS for OpenEdge, provides HTTP sessions as the default method for preserving session and contextual information.

Since information in HTTP sessions can be accessible to a cluster of PAS for OpenEdge instances, HTTP sessions are useful for load balancing. (See Load balancing options for PAS for OpenEdge on page 69.)

Note: Information about the third party technologies mentioned in this section is very general. For more detailed information, do a Web search for key terms: HTTP sessions, JSESSIONID,
Chapter 5: HTTP sessions

Administrative considerations
To enable HTTP sessions for the APSV transport (HTTP session is automatically enabled for SOAP, REST and WEB), both `enableRequestChunking` and `useHTTPSessions` must be activated in the `/conf/openedge.properties` file. For example:

```
enableRequestChunking=1
useHTTPSessions=1
```

HTTP sessions are enabled by default in PAS for OpenEdge instances and there is nothing you need do to configure them. However, you can check these settings in an instance's `openedge.properties` file to ensure that HTTP sessions are enabled.

**Note:** If you want to use HTTP sessions, the ABL client must be OpenEdge 11.6 or a later release. Open Clients (.NET and JAVA) are not supported.

Notice that chunking must be enabled as well as HTTP sessions. Chunking is an efficient method for sending a large amount of data in a request or response when the size of the content is not known. Chunking also that all the data gets transmitted in a single post operation in the context of a single session to a single server. You can use chunking if HTTP sessions is disabled (if, for example, you only have one instance and do not require load balancing). However, you will get a startup error if chunking is disabled and HTTP sessions is enabled.

When load balancing is implemented using clusters of PAS for OpenEdge instances, HTTP sessions allows session context information to be available to all the nodes in the cluster. Therefore a single session between a client and a Web application can be maintained over any number of instances in the cluster. See Clusters and sticky sessions on page 67 and Tomcat load balancing on page 73 for more information.

**Note:** You can use the TCMAN `sessions` action to display how many sessions are active for a Web application.

Programmatic considerations
The Web application determines when to start and stop a session. Also, it not only determines what contextual information will be stored in the session, it determines how that information will be stored.

**Note:** The Spring Security framework built in to PAS for OpenEdge saves client login state information to HTTP sessions.

By default, when a Web application running on a PAS for OpenEdge instance decides to initiate a session with a client, it creates a JSESSIONID cookie that identifies the session. The cookie is sent to the client. Every time the client runs a request to that application, the cookie gets sent back to the PAS for OpenEdge instance, whose job is to identify and retrieve the stored data that is associated with the session id.

Context information required for the session is automatically saved in HTTP sessions. Again, it is up to the application to determine what needs to be saved and how to save it. Context data itself can be saved to and retrieved from HTTP sessions. Or you can save the context information externally and just save a reference to it in HTTP sessions. The latter option is more efficient when there is a large amount of data associated with the session.
In ABL for example, the SERVER-CONNECTION-CONTEXT attribute of the SESSION system handle is commonly used to set context information. If the context information can be expressed as a string of less than 32K, the attribute can be set to that value. The 32K maximum may be even less, depending on the other limitations imposed by your particular configuration. However, the more data you store in SERVER-CONNECTION-CONTEXT the more performance is likely to suffer. Therefore, the recommended practice is to store the context information in a database and use the SERVER-CONNECTION-CONTEXT attribute as a reference to the context information.

The implementation of how to store context varies widely, and depends on application requirements. For more information on how to use the SERVER-CONNECTION-CONTEXT attribute see Progress Application Server for OpenEdge: Application and Migration Development Guide and OpenEdge Development: ABL Reference.

See also
Display Web application HTTP sessions (sessions) on page 131

Clusters and sticky sessions

HTTP sessions is an enabling technology for load balancing. With HTTP sessions, session information can be shared with any PAS for OpenEdge instance in a defined cluster of instances. Therefore, any instance in the cluster can handle subsequent client requests within the correct context.

Instances in a cluster must all be created from a common core PAS for OpenEdge installation. ($CATALINA_HOME). To add an instance to a cluster, you must turn on the cluster property in the /conf/server.xml file of the instance. To avoid corrupting server.xml use the TCMAN utility to update the property. For example:

```
/bin/tcman.sh feature Cluster=on
```

Sticky sessions

HTTP sessions can be used to contain context information whether sticky sessions are enabled or disabled.

Sticky sessions, another industry standard for Web servers, binds a session to a particular instance. In other words, after an initial client request is satisfied by a Web application running in a PAS for OpenEdge instance, subsequent requests are routed to the same Web application running on the same instance. You would want sticky sessions enabled when:

- The Web application is storing state information about the session
- The Web application launches persistent procedures, since the memory space that persistent procedures run in cannot be shared by multiple instances.
In a load balancing configuration, sticky sessions are usually enabled or disabled by configuring the Web server. In an Apache Web server, you enable sticky sessions by adding a `route` property to `BalancerMember` and setting the `stickeysession` property. For example:

```xml
#Load balancer Proxy settings (sticky sessions)
<Proxy "balancer://mylbgroup">
    BalancerMember "http://172.16.21.39:11600" route=oepas1
    BalancerMember "http://172.16.21.39:11610" route=oepas2
    BalancerMember "http://172.16.21.39:11620" route=oepas3
</Proxy>
ProxyPass "/test" "balancer://mylbgroup"
stickysession=JSESSIONID|jsessionid nofailover=On
ProxyPassReverse "/test" "balancer://mylbgroup"
```

When sticky sessions are not enabled on an Apache Web server, the instances in the configuration are specified by their IP addresses only and the `stickeysession` property is not set. For example:

```xml
#Load balancer Proxy settings (session free)
<Proxy "balancer://mylbgroup">
    BalancerMember "http://172.16.21.39:11600"
    BalancerMember "http://172.16.21.39:11610"
    BalancerMember "http://172.16.21.39:11620"
</Proxy>
ProxyPass "/test" "balancer://mylbgroup"
ProxyPassReverse "/test" "balancer://mylbgroup"
```

**Note:**

- The Tomcat servlet container in PAS for OpenEdge does not allow Web applications to change a session to a sticky session.
- Sticky sessions can have a negative impact on performance in a load balancing configuration since client requests can be bound to a single node among all available nodes.
Load balancing options for PAS for OpenEdge

If you are familiar with configuring classic WebSpeed, you will find that load balancing (distributing client requests among multiple servers) is quite different in PAS for OpenEdge. In PAS for OpenEdge, there is no NameServer or ubroker.properties file to implement load balancing. Instead, load balancing is accomplished through industry standard HTTP options.

This section describes three of the most popular load balancing options for PAS for OpenEdge. It includes an overview of configuration tasks for each option, and, where applicable, describes the strengths and weaknesses of the option. There are many more 3rd party load balancing products available. It is not possible to recommend the best load balancing option for your application.

For details, see the following topics:

- Apache proxy host load balancing
- Tomcat load balancing
- Amazon load balancing

Apache proxy host load balancing

With Apache proxy host load balancing, an Apache HTTP Server (also known as Apache) can direct requests to a particular PAS for OpenEdge instance. The following figure illustrates this load balancing configuration.
Apache works as a proxy re-directing HTTP requests to a URL that is unique to each instance. This method requires no configuration of the PAS for OpenEdge instance, all of the load balancing is done by the Apache server. The available PAS for OpenEdge instances are defined in an
apache_install_dir/conf/httpd.conf file.

Note: The disadvantage in this type of load balancing is that the Apache server does not monitor the running status of instances. If an instance crashes or is otherwise unavailable, the Apache load balancer continues to try to send requests to the unavailable instance and requests will be lost. Compare with Tomcat load balancing on page 73, where the running status of instances is monitored.

See also
Requirements for Apache proxy load balancing on page 70
Configuring the Apache server for proxy load balancing on page 71

Requirements for Apache proxy load balancing

The requirements for implementing Apache proxy load balancing are:

1. A properly configured Apache HTTP Server.
   You configure a load balancing Apache HTTP Server (commonly referred to as Apache) with a list of available PAS for OpenEdge instances. In addition, you specify whether you want subsequent requests to return to the same instance (session-managed, also known sticky sessions in Apache documentation), or to any available instance (session-free).
   If security is an issue, (that is, if you are running Web application available to the public), the recommendation is to install Apache in a DMZ with the PAS for OpenEdge instances running behind a second firewall.

2. Properly configured PAS for OpenEdge instances.
   Configuration of instances is contingent on whether sticky sessions are enabled on the Apache load balancer. All the instances must be configured to match the session type of the load balancer.

See also
Configuring the Apache server for proxy load balancing on page 71
Configuring PAS for OpenEdge instances for proxy load balancing on page 72
Configuring the Apache server for proxy load balancing

Configuring an Apache HTTP Server to implement load balancing involves updating the `apache_install_dir/conf/httpd.conf` file to:

- Specify the proxy modules to load
- Reference the PAS for OpenEdge instances
- Enable or disable sticky sessions

**Note:** In addition to configuring the Apache server, you must configure each PAS for OpenEdge instance. See Configuring PAS for OpenEdge instances for proxy load balancing on page 72

### Loading proxy modules

Proxy modules set up the Apache Web server as a proxy to handle requests to multiple PAS for OpenEdge instances. The modules are loaded by un-commenting `LoadModule` directives in the Apache Web server's `httpd.conf` file. The `LoadModule` directives include:

- `LoadModule proxy_module modules/mod_proxy.so`
- `LoadModule proxy_connect_module modules/mod_proxy_connect.so`
- `LoadModule proxy_http_module modules/mod_proxy_http.so`
- `LoadModule slotmem_shm_module modules/mod_slotmem_shm.so`
- `LoadModule bmethod_byrequests_module modules/mod_lbmethod_byrequests.so`

**Note:** Depending on which version of the Apache server you are using, there may be additional modules required. See the Apache Web site (http://httpd.apache.org/) for documentation on your version of the Apache server.

### Referencing instances and configuring session types

Update Apache’s `httpd.conf` file to include a load balancer section that lists the PASOE instances in the load balancing group. This can be configured two different ways:

- **Session free** — any request can be sent to any PAS for OpenEdge instance.
- **Sticky session** — once a session is started all requests return to the same PAS for OpenEdge instance until the session is terminated

If there is no need for client requests to return the the same PAS for OpenEdge instance (i.e. maintaining server-side context is not required), your entries in `httpd.conf` should resemble the following example:

```bash
#Load balancer Proxy settings (session free)
<Proxy "balancer://mylbgroup">
    BalancerMember "http://172.16.21.39:11600"
    BalancerMember "http://172.16.21.39:11610"
    BalancerMember "http://172.16.21.39:11620"
</Proxy>
ProxyPass "/test" "balancer://mylbgroup"
ProxyPassReverse "/test" "balancer://mylbgroup"
```
Notice that, in the example above, the instances where Apache can refer requests are grouped as `//mylbgroup` and the URL of these instances (the `BalancerMember` properties) are specified.

If it is necessary for sequential requests to go to the same instance, you enable sticky sessions by adding a `route` property to `BalancerMember` and by enabling JSESSIONID cookies. For example:

```xml
#Load balancer Proxy settings (sticky sessions)
<Proxy "balancer://mylbgroup">
  BalancerMember "http://172.16.21.39:11600" route=oepas1
  BalancerMember "http://172.16.21.39:11610" route=oepas2
  BalancerMember "http://172.16.21.39:11620" route=oepas3
</Proxy>
ProxyPass "/test" "balancer://mylbgroup" stickysession=JSESSIONID|jsessionid
noopalover=On
    ProxyPassReverse "/test" "balancer://mylbgroup"
```

In the example above notice that:

- The `route` property must match the `jvmroute` value in the instance's `server.xml` file.
- The `stickysession` property enables the JSESSIONID cookie, which is the mechanism used to identify the session.

**Note:** For more information about session configurations in the `httpd.conf` file, see the Apache Web site ([http://httpd.apache.org/](http://httpd.apache.org/)) for documentation on your version of the Apache server.

**See also**

Configuring PAS for OpenEdge instances for proxy load balancing on page 72

**Configuring PAS for OpenEdge instances for proxy load balancing**

For any type of load balancing, chunking and HTTP sessions must be enabled, and they are both enabled by default.

To verify that chunking and HTTP sessions are enabled, check the `useHTTPSessions` and `enableRequestChunking` properties in the `/conf/openedge.properties` file for each PAS for OpenEdge instance. When enabled, these properties are set as shown in bold in the following example:

```properties
[instance_name.ROOT.APSV]
    adapterEnabled=1
    enableRequestChunking=1
    useHTTPSessions=1
```

**Note:** You can also run the following commands to check the current values:

- `oeprop instance_name.ROOT.APSV.enableRequestChunking`
- `oeprop instance_name.ROOT.APSV.useHTTPSessions`
If sticky sessions are enabled on the Apache server, you are done. But if the Apache server is configured for session-free load balancing (where subsequent requests do not return to the same instance), you must configure the PAS for OpenEdge instances as follows:

1. Enable `Cluster` property in each instance’s `/conf/server.xml` file. You can use the `feature` action of the TCMAN command-line utility to set this property. For example:
   
   ```
oepas1/bin/tcman.sh feature Cluster=on
   ```

2. Set the `jvmRoute` alias to blank.

   The `jvmRoute` alias is controlled by the `psc.as.alias` value and can be set to blank with TCMAN (rather than by manually editing `/conf/server.xml`) as follows:
   
   ```
tcman config psc.as.alias=
   ```

   **Note:** The `jvmRoute` alias translates to the route value in the Apache server’s `http.conf` file. See Configuring the Apache server for proxy load balancing on page 71 for more information.

3. Open the `/conf/context.xml` file in each instance and comment out the `Manager` section. For example:
   
   ```
   <!-- <Manager
       maxActiveSessions="-1"
       pathname=""
       processExpiresFrequency="6"
       maxInactiveInterval="${psc.as.session.timeout}"
       sessionIdLength="22" >
   <Manager>  -->
   ```

See also

Configuring the Apache server for proxy load balancing on page 71

---

**Tomcat load balancing**

Tomcat load balancing is achieved by designating a PAS for OpenEdge instance that does nothing but refer requests from a Web server to instances that can process them. Tomcat is built into PAS for OpenEdge. The following figure illustrates this load balancing configuration.
Some advantages of this configuration, compared to Apache proxy host load balancing on page 69, are:

- The PAS for OpenEdge instance that is designated as a load balancer (lb in the figure above) is aware of the runtime status of each instance in its group and will not forward a request to an instance that is down.
- Tomcat load balancing allows you to configure a PAS for OpenEdge instance as a monitor of runtime statistics for each instance in the load balancing group (the status instance in the figure above).

See also
Requirements for Tomcat load balancing on page 74
Configuring PAS for OpenEdge for Tomcat load balancing on page 75

Requirements for Tomcat load balancing

The following are the requirements to implement Tomcat load balancing:

1. A properly configured Web server.
   
   If security is an issue, (that is, if you are running Web application available to the public), the recommendation is to install the Web server in a DMZ with the PAS for OpenEdge instances running behind a second firewall.

2. A PAS for OpenEdge instance that is dedicated to load balancing.
   
   You should not deploy any Web applications to the load balancer.

   
   The workers.properties file specifies the existence of a load balancer, the workers (instances) that are available to the load balancer as servers that can handle requests, an optional status worker that monitors the runtime status of the other workers.
   
   You can create a workers.properties file with the TCMAN workers action.

See also
Create a Tomcat worker configuration file (workers) on page 152
**Configuring an Apache Web server for Tomcat load balancing**

Apache Web server configuration for Tomcat load balancing involves enabling the AJP13 protocol for communicating with the load balancer and adding a number of definitions that reference the load balancer.

**Note:** Tomcat load balancing can be done with Web servers other than Apache. However, only Apache configuration is described here because Apache is arguably the most common Web server, and it is not practicable to describe the configuration details of all the available Web servers.

Basically, you do the following:

1. **Open Apache's** `/conf/httpd.conf` file and un-comment the `LoadModule` directives in the Apache Web server's `httpd.conf` file that enable AJP13. The `LoadModule` directives include:
   - `LoadModule proxy_ajp_module modules/mod_proxy_ajp.so`
   - `LoadModule proxy_balancer_module modules/mod_proxy_balancer.so`

   **Note:** Depending on which version of the Apache server you are using, there may be additional modules required. See the Apache Web site (http://httpd.apache.org/) for documentation on your version of the Apache server.

2. Add the following definitions to `/conf/httpd.conf`:

   ```
   #workers.properties load balancing config
   LoadModule jk_module apache_install_dir/modules/mod_jk.so
   JkWorkersFile apache_install_dir/conf/workers.properties
   JkShmFile apache_install_dir/logs/mod_jk.shm
   JkLogFile apache_install_dir/logs/mod_jk.log
   JkLogLevel info
   JkLogStampFormat "[%a %b %d %H:%M:%S %Y]"
   JkMount /* jklb
   ```

3. Use TCMAN to enable the AJP13 port in the `/conf/server.xml` file.
   For example:

   ```
   instance/bin/tcman feature AJP13=on
   ```

   Apache now passes all requests to a PAS for OpenEdge load balancing instance via its AJP13 port.

   **Note:** The PAS for OpenEdge load balancing instance is referenced in the example above as `jklb(JkMount /* jklb)`. The actual instance name must be `lb`.

---

**Configuring PAS for OpenEdge for Tomcat load balancing**

Configuring PAS for OpenEdge to enable Tomcat load balancing involves:
1. Creating an instance (lb) that is dedicated to load balancing.

You can use the TCMAN create action to instantiate an instance that must be named either lb or jklb. No other configuration is required, but note that you should not deploy any Web applications to the lb instance.

2. Optionally creating an instance (status) to gather metrics on the instances in the load balancing group.

You can use the TCMAN create action to instantiate an instance that must be named status or jkstatus. No other configuration is required, but note that you should not deploy any Web applications to the status instance.

Also note that the lb automatically keeps track of the runtime status of instances in its load balancing group. It will not forward a request to an instance that has crashed or is not running for some other reason. Therefore, the status instance is only required if you want to monitor performance data about the instances in the load balancing group.


The worker.properties file is a Tomcat feature that defines the load balancing instance (lb), the status instance, and the instances that handle requests status.

You can create a preliminary worker.properties file using the TCMAN (or PASMAN) workers action. The action gathers information on all instances registered to the local PAS for OpenEdge core server (CATALINA_HOME). Note that there must be an instance named lb or jklb that will function as the load balancer.

The workers action only includes local instances. If you have instances on multiple machines, you must:

1. Run the TCMAN workers action on each machine.
2. Combine the workers.properties file from each machine into a single workers.properties file.
3. Comment out the workers.common.host property.
4. Add a workers.instance_name.host=host_name property to each instance.

Note:

- The TCMAN workers action references the content of the $CATALINA_HOME/extras/workers.template file when it builds a preliminary worker.properties file. You can modify the workers.template file to include settings that are typical in your local configuration.

- The preliminary worker.properties is created in the temp directory of the CATALINA_HOME server (for example, $DLC/servers/pasoe/temp). Use this file as a starting point; you may need to modify parameter values to better suit your particular application configuration.

See also

Create an instance (create) on page 132
Create a Tomcat worker configuration file (workers) on page 152
Amazon load balancing

Amazon Web Services offers an Elastic Load Balancing (https://aws.amazon.com/elasticloadbalancing/) product that distributes incoming application across multiple Amazon EC2 instances in the cloud. It also offers an Auto Scaling (https://aws.amazon.com/documentation/autoscaling/) product that automatically monitors the EC2 instances and manages them according to policies that you can define. You can use these products together to create a cloud-based load balancing configuration for PAS for OpenEdge.

See also
Configuring Elastic Load Balancing and Auto Scaling on page 77

Configuring Elastic Load Balancing and Auto Scaling

This topic is an overview of configuring Amazon Elastic Load Balancing and Auto Scaling as it applies to PAS for OpenEdge. For more information about the Amazon products, see https://aws.amazon.com/documentation/elastic-load-balancing/ and https://aws.amazon.com/documentation/autoscaling/.

The following table lists the relevant terms applied to the components of these products.

Table 7: Elastic Load Balancing and Auto Scaling components

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual Private Cloud (VPC)</td>
<td>A private network of your application machines, which is your internal, &quot;safe&quot; network away from the internet. Access can be limited to specific incoming IP addresses or ranges.</td>
</tr>
<tr>
<td>Amazon Machine Image (AMI)</td>
<td>A copy of a virtual machine that, when launched, will automatically start the processes that support the load balanced application.</td>
</tr>
<tr>
<td>Elastic Load Balancer (ELB)</td>
<td>An internet facing load balancer that redirects work to the instances running in a VPC. Monitors instance health and redirects using a proprietary cookie (AWS_ELB).</td>
</tr>
<tr>
<td>Launch Configuration</td>
<td>A definition of the configuration of the AMI you want to launch automatically when scaling up.</td>
</tr>
<tr>
<td>Auto Scaling Groups</td>
<td>A definition of a group of AMI's that specifies their ELB and the scaling policy (how and when scaling up or down occurs).</td>
</tr>
</tbody>
</table>
The basic steps for creating an Elastic Load Balancing and Auto Scaling configuration are:

1. Create a VPC (or use the default) which includes:
   - defining a network mask range for your private network
   - defining a security group that limits access to the private network

2. Create the AMIs.
   Each AMI must have a PAS for OpenEdge instance installed, and the instance must be configured to start when the operating system launches. Note that the instances can be configured to use the same ports since they are running on different hosts and therefore will have unique IP addresses in the VPC.

3. Create an ELB, enabling:
   - an internet facing scheme
   - port forwarding (for example, forward port 80 to the PAS for OpenEdge port 8810)
   - sticky sessions (using the AWS_ELB cookie) or cookie timeout.
   - cross-zone load balancing
   - log timing
   - idle timeout
   - a Health Check

4. Create a Launch Configuration that specifies:
   - an AMI
   - the instance size (CPUs, memory, and disk space)
   - the VPC
   - the security group

5. Create an Auto Scaling Group that specifies:
   - the ELB
   - the VPC
   - the maximum and minimum number of copies of the AMI that can be launched
   - a Health Check configuration
   - the scaling policy (what metric threshold causes scaling up or down)
   - the tag to apply at instance launch

Note: The preceding steps are just an overview of the procedures required to implement Amazon load balancing. See the Amazon documentation for more complete information.
Server security

This section provides information about security configurations for instances of PAS for OpenEdge. Generally speaking, server security support includes standard Tomcat server functionality, the Spring Security Framework, plus OpenEdge security features and customizations.

For details, see the following topics:

- Development server and production server security issues
- HTTPS support
- Configuring PAS for OpenEdge for SSL/TLS
- The Spring Security Framework in PAS for OpenEdge
- JVM security manager
- Realms and roles
- Remote access filters

Development server and production server security issues

The Progress Application Server (PAS) for OpenEdge is a Web application server that is available as either a development server or a production server product.
The difference between a development server and a production server is largely a matter of security. Typically, a server in a development environment allows unrestricted access to and control by a user or group of users. A production server, on the other hand, is configured to restrict access to authorized users and to limit control to system administrators. For example, in a development environment anyone might be allowed to shut down the server, whereas, in a production environment, only an administrator with appropriate privileges would be allowed to stop a running server.

Some of the security customizations in the production server version of PAS for OpenEdge are:

- **Removal of manager and host-manager Web applications** — These are default Tomcat applications that enable remote online administration. These applications are archived in the $CATALINA_HOME/extras directory and can be redeployed to enable their functionality.

- **Replacement of the default Tomcat webapps/ROOT application** — Replaced by a ROOT application that specifically supports Progress applications, including application security. The Tomcat ROOT application is archived in the $CATALINA_HOME/extras directory.

- **Auto-deployment turned off by default** — Prevents the deployment of WAR files that are maliciously or erroneously copied to the server. If turned on, the server automatically deploys any new or updated WAR files in its Web application directory. (Note however that the server will automatically unpack WAR files when they are legitimately deployed using TCMAN or standard Tomcat utilities.)

- **Shutdown port disabled for UNIX** — Prevents unauthorized stopping of the server.

  Note that a shutdown port is optional for servers running on UNIX systems. However, you must specify a shutdown port when you create a server that runs on Windows systems. The TCMAN utility supports specifying shutdown ports with the \(-s\) option to the create action.

- **JMX remote access not enabled** — Although PAS includes JMX and JConsole support for server management, the default is for local access only.

- **Web crawler filtering enabled** — Prevents server being overloaded by sessions initiated by web crawlers.

  **Note:** In PAS for OpenEdge, a ROOT application that specifically supports OpenEdge application security replaces the default Tomcat ROOT application. The Tomcat ROOT application is archived in the $CATALINA_HOME/extras directory.

### HTTPS support

The Progress Application Server (PAS) allows you to configure an HTTPS port when you create an instance of a PAS server. The TCMAN utility supports specifying HTTPS ports with the \(-P\) option to the create action.

For example, the following instance is created with an HTTP port of 8501 and an HTTPS port of 8601:

```
$: $CATALINA_HOME/bin/tcman.sh create -p 8501 -P 8601 /psc/acme1
Server instance acme1 created at /psc/acme1
```

In PAS (and in Tomcat as well) HTTPS port support is provided by the Secure Socket Layer (SSL) or Transport Layer Security (TLS) implementation of the Java Secure Socket Extension (JSSE).
Important:

On a production server, you must replace the default SSL/TLS Server Certificate with a fully qualified certificate from a Certificate Authority (CA).

Server access via HTTPS is dependent on having an SSL/TLS Server Certificate. To support initial configuration and testing, PSC includes a self-signed server certificate in a Java keystore. (A key store is basically a file that contains the encrypted keys required for encoding and decoding information.) The PSC keystore is formatted according to the PKCS12 format and is maintained using the Java Keytool utility.

However, the default server certificate should only be used on a non-production server, or as a temporary certificate for setting up SSL or TLS on a production server. Using the default certificate on a fully functional and deployed production server, is a serious security risk.

See Configuring PAS for OpenEdge for SSL/TLS on page 81 for more information.

Configuring PAS for OpenEdge for SSL/TLS

Secure Sockets Layer (SSL) and the newer Transport Layer Security (TLS) are standard protocols that are used to implement the encryption required for HTTPS communication. Both protocols employ key pairs (a public key and a private key) to secure logins, financial transactions, and many other types of data transfer between web servers and clients. Basically, the public key is used by clients to encrypt data, but only a server that has a valid private key can decrypt the data.

Note:

TLS evolved from and superceded SSL. TLS is considered to be more efficient and secure than the older SSL. Even though TLS and SSL are separate and distinct protocols, the PAS for OpenEdge configuration process is the same for both. Therefore we use the term SSL/TLS.

Also note that most browsers concurrently support some version of both protocols. For more information, do a web search for SSL and TLS.

The following is an overview of SSL/TLS configuration in PAS for OpenEdge:
See the following topics for more details about SSL/TLS configuration:

- **Obtaining a certificate from a Certificate Authority** on page 82
- **Configuring a PAS for OpenEdge instance for SSL/TLS** on page 85
- **Configuring clients for SSL/TLS** on page 86

### Obtaining a certificate from a Certificate Authority

This section describes how to generate a key file that you can submit to a third party Certificate Authority (CA) to obtain an SSL/TLS certificate. It also shows how to import the certificate from the CA to your PAS for OpenEdge instance.

1. Start PROENV and change directory to `OpenEdge-Install-Dir\keys\requests`.
   
   For example:
   ```bash
   proenv> cd C:\Progress\OpenEdge\keys\requests
   ```
Note:
On Windows, you must start PROENV with the Run as administrator option. For example:

2. Generate a 2048-bit private key (certificate_name.pki) and a public key (certificate_name.pki0) file.

For example:

```
proenv>pkiutil -keysize 2048 -newreq MyCert
```

Note: PKIUTIL can generate key sizes of 512, 1024, or 2048 bits.

PKIUTIL prompts for a PEM pass phrase (which you must compose) with output similar to the following:

```
Loading 'screen' into random state - done
Generating a 2048 bit RSA private key
...+
........................+++ writing new private key to'C:\Progress\OE\102a\dlc/keys/requests/MyCert.pki'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
----
```

After you enter and verify the pass phrase, the private key file (MyCert.pki in this example) is generated.

Important: Make a note of the PEM pass phrase. It will be required when you import the certificate returned from a CA to create a keystore in Step 5 on page 84.
Then, PKIUTIL prompts for the specific name of the system (Server DNS name) and other information that will be included in the certificate request. The information is incorporated into a Distinguished Name (DN). If a single period '. ' is entered, the field will be left blank. For example:

```
-----
Country Name (2 letter code) [US]:US
State or Province Name (full name) []:Massachusetts
Locality Name (eg, city) []:Bedford
Organization Name (eg, company) []:ACME1
Organizational Unit Name (eg, section) []:.
Server DNS name []:bedford.acme1.com
```

Thereupon, a public key file (MyCert.pk10 in this example) is created. The public key file is what you use to request a new digital certificate from the CA.

3. Submit MyCert.pk10, the public key file, to a CA in order to request an SSL certificate.

   The CA returns both a private and a public (or ROOT) SSL certificate. The certificates are files, usually with either a .crt or a .cer extension. In this example, we'll call the files MyCertPriv.cer and MyCertPub.cer

4. When the SSL certificates are received from the CA, copy them to the OpenEdge-Install-Dir\keys\requests directory.

5. Generate a Privacy Enhance Mail (.pem) formatted file from the private SSL certificate (named MyCertPriv.cer in these examples) obtained from a CA.

   A PEM file is an encrypted file that contains key store information. You use the OpenEdge PKIUTIL command-line utility to generate the PEM file. You can find more information about the syntax and usage of PKIUTIL in OpenEdge Getting Started: Installation and Configuration.

   a) In PROENV, change directory to the PAS for OpenEdge instance's/conf directory.

   For example:

   ```
   proenv> cd C:\MyInstance\conf
   ```

   b) Use the -import option of PKIUTIL to generate the PEM file from the private SSL certificate.

   For example:

   ```
   proenv> pkiutil -import MySSLPrivKey
   OpenEdge-Install-Dir\keys\requests\MyCertPriv.cer
   ```

   **Note:** MySSLPrivKey is the stem filename of the PEM file that will be generated from MyCertPriv.cer.

   c) When prompted enter the password you used when you created the keystore (i.e. the .pk1 file) in Step 2 on page 83.

   PKIUTIL creates a file with a .pem extension (MySSLPrivKey.pem in this example) in the OpenEdge-Install-Dir\keys\ directory.
Configuring a PAS for OpenEdge instance for SSL/TLS

Essentially, configuring a PAS for OpenEdge instance for SSL/TLS involves updating the default Tomcat keystore file in your PAS for OpenEdge instance with the information from an SSL/TLS certificate obtained from a Certificate Authority (CA).

**Note:** You will need the password that you used when you ran PKIUTIL to create a public key certificate request file (MyCert.pk10 in the example procedure described in the Obtaining a certificate from a Certificate Authority on page 82 topic).

**To update a PAS for OpenEdge keystore:**

1. Make a backup copy of the default Tomcat keystore file located in your PAS for OpenEdge instance's /conf directory.

   For example:
   ```
   proenv> cd C:\MyInstance\conf
   proenv> copy tomcat-keystore.p12 tomcat-keystore.p12.original
   ```

2. From the instance's /conf directory, use the `sslc` command to export the information from the PEM file to the Tomcat keystore.

   **Note:** The `sslc.exe` executable is an OpenSSL (https://www.openssl.org/) command-line utility that is included in `OpenEdge_install_dir/bin`.

   For example:
   ```
   proenv> cd C:\MyInstance\conf
   proenv> sslc pkcs12 -export C:\Progress\OpenEdge\keys\V.pem -out tomcat-keystore.p12 -name mysslprivkey
   ```

   **Important:**
   Avoid using upper-case letters in the filename specified after the `-name` parameter. The `sslc` automatically converts the name to lower-case in the alias entry (Alias name: mysslprivkey) in the keystore. Using upper-case letters could cause a mismatch when you update the /conf/catalina.properties file. See Step 5 below.

3. When prompted enter the password you used when you created the keystore (i.e. the .pk1 file). See Obtaining a certificate from a Certificate Authority on page 82.

4. Use the `sslc` command to verify what is now in the Tomcat keystore.

   For example:
   ```
   sslc pkcs12 -info -in ./tomcat-keystore.p12
   ```
5. Update the instance's /conf/catalina.properties file to update the psc.as.https.keypass and psc.as.https.keyalias properties in the JSSE keystore section.

For example:

```
# JSSE keystore used by server.xml for its server key & certificates
psc.as.https.keypass=your_password
psc.as.https.keyalias=mysslprivkey
psc.as.https.storeType=PKCS12
```

**Important:** The value for psc.as.https.keyalias must match the value generated by sslc, which is always expressed in lower-case letters.

6. Restart the instance and test.

You can restart the instance using the TCMAN stop and start actions. Once the instance has restarted, test the connection and authentication configurations from a browser.

**Note:** You may need to import the CA certificate to the browser.

---

### Configuring clients for SSL/TLS

In OpenEdge, you can use the certutil command to create the files that supply the necessary keys for connecting to clients.

To generate the key files and copy them to a server:

1. On a PROENV command line, change your working directory to the `OpenEdge-Install-Dir\keys\requests`.

   This should be the directory where you copied the SSL/TLS certificates, which is described in *Obtaining a certificate from a Certificate Authority* on page 82.

2. Run certutil to import both the public and private SSL/TLS certificates to the `OpenEdge-Install-Dir\certs` directory.

   The following is an example with certificates named MyCertPriv.cer and MyCertPub.cer in the C:\Progress\OpenEdge\keys\requests directory:

   ```
   proenv> certutil -import C:\Progress\OpenEdge\keys\requests\MyCertPriv.cer
   Importing trusted certificate to alias name: 9792edd3
   proenv> certutil -import C:\Progress\OpenEdge\keys\requests\MyCertPub.cer
   Importing trusted certificate to alias name: 22c9eb58
   ```

**Notice:**

- The -import parameter requires the fully qualified pathname of the certificate file.
- The key file alias names are exactly eight hexadecimal characters in length and have a .0 (dot-zero) file extension. However, keep in mind that your files will not have the same filenames that are shown in this example.
3. Verify that the key files created in the previous step exist in the `OpenEdge-Install-Dir\certs` directory.

4. Copy the key files to the client.
   
   Every client that will connect to the server must have a copy of the key file that was generated by the import.

5. Test the client.

### The Spring Security Framework in PAS for OpenEdge

The Progress Application Server (PAS) for OpenEdge employs the Spring Security framework to provide enterprise-level application security with regard to:

- **Authentication** — verification that username and password combinations are valid.
- **Authorization** — enforcing access control (user accounts are granted permissions according to their assigned roles).

Spring Security is a Java-based framework provided in the `/common/lib` directory of the core PAS server. For general information about Spring Security, see [http://www.springsource.org](http://www.springsource.org). For reference and API information, see [http://static.springsource.org/spring-security/site/reference.html](http://static.springsource.org/spring-security/site/reference.html)

### JVM security manager

When you run Tomcat, each instance of the server runs in the context of its own individual JVM whose security can be managed with the Java Security Manager.

PAS is enabled with a default configuration of the standard JVM Security Manager. The JVM security is configured in the `/conf/catalina.policy` file of a given PAS instance.

The `catalina.policy` file uses Permission classes, either the built-in JDK classes or customized classes, to control what actions a class can take. The following are just a few examples of the built-in classes:

- `java.io.FilePermissions`: controls access to read, write, delete, and other actions for files and directories
- `java.net.SocketPermissions`: controls network access through sockets
- `java.security.SecurityPermissions`: controls access to security methods


### Realms and roles

A realm is a collection of usernames and passwords, and it includes the roles associated with each of those users. PAS can support one or more realm definitions. A role controls the amount of access a given group of users has. All authorizations in The Progress Application Server are role-based; access privileges cannot be granted on a user-by-user basis.
PAS retains the roles defined in Tomcat's default MemoryRealm, but also adds a set of PAS roles that map to the Tomcat roles and allows for consistency across the product. For example, ROLE_PSCAdmin allows unrestricted administrator access, and it maps to Tomcat's admin-gui, admin-script, manager-gui, manager-script, and manager-status roles.

Managing users and roles for Manager and Host Manager applications

For user authentication, the Progress Application Server comes configured for use with Tomcat's MemoryRealm. The MemoryRealm accesses user authentication information stored in an XML file, conf/tomcat-users.xml. You can replace the MemoryRealm with a different type of realm, such as JDBC or LDAP, and you can have multiple realms of different types configured at the same time.

**Note:** The MemoryRealm configuration is not intended for use in a production environment. For production, you should replace the MemoryRealm with a stronger source of user authentication.

PAS retains the roles defined in Tomcat's default MemoryRealm, but PAS also includes a new set of roles that include the Tomcat roles. The following table lists the new roles and what existing Tomcat roles they map to, if any:

**Table 8: PAS roles mapped to Tomcat roles**

<table>
<thead>
<tr>
<th>PAS role</th>
<th>Includes Tomcat Manager and Host Manager roles</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROLE_PSCAdmin</td>
<td>admin-gui, admin-script, manager-gui, manager-script, manager-jmx, manager-status</td>
<td>Unrestricted administration access. The Tomcat admin roles allow access to the HTML GUI and text interfaces of the Host Manager application, and the manager roles allow access to the HTML GUI, text interface, JMX proxy, and status pages of the Manager application.</td>
</tr>
<tr>
<td>ROLE_PSCOOper</td>
<td>admin-script, manager-script, manager-status</td>
<td>Restricted operator administration access. The Tomcat roles allow access to the text interfaces of the Host Manager and Manager applications as well as the status pages in the Manager application.</td>
</tr>
<tr>
<td>ROLE_PSCUser</td>
<td>none</td>
<td>Non-priviliged access</td>
</tr>
<tr>
<td>ROLE_PSCNone</td>
<td>none</td>
<td>No access, for user account authorization testing</td>
</tr>
</tbody>
</table>

**Note:** If you use a realm(s) other than the default MemoryRealm, that realm must have the roles listed in the table above.

PAS comes with three default test accounts configured in conf/tomcat-users.xml, as described in the following table:
Table 9: PAS default user accounts

<table>
<thead>
<tr>
<th>Username</th>
<th>Password</th>
<th>Role(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>tomcat</td>
<td>tomcat</td>
<td>ROLE_PSCAdmin, ROLE_PSCOper, ROLE_PSCUser</td>
</tr>
<tr>
<td>tcuser1</td>
<td>tcuser</td>
<td>ROLE_PSCUser</td>
</tr>
<tr>
<td>tcuser2</td>
<td>tcuser</td>
<td>ROLE_PSCNone</td>
</tr>
</tbody>
</table>

The format for users entries in conf/tomcat-users.xml is shown below, with the tomcat default user given as an example:

```xml
<user username="tomcat" password="tomcat"
     roles="ROLE_PSCAdmin,ROLE_PSCOper,ROLE_PSCUser"/>
```

## Remote access filters

The Progress Application Server (PAS) is configured with filters that allow you to deny access based on IP addresses or host names. These filters are implemented as valves in the PAS instance's /conf/server.xml file. The following snippet from the default server.xml shows that the filters are set to allow access by anyone:

```xml
<Valve className="org.apache.catalina.valves.RemoteHostValve"
     allow=".*"/>
<Valve className="org.apache.catalina.valves.RemoteAddrValve"
     allow=".*"/>
```

The RemoteHostValve and RemoteAddrValve filters can take two attributes:

**allow**

A comma-delimited list of regular expressions that a client must match in order to be allowed access.

**deny**

A comma-delimited list of regular expressions that, when matched, excludes a client from access.
Web application security configurations

In Progress Application Server for OpenEdge, the Spring Security framework implements Web application security.

Spring Security requires a separate authentication provider plug-in for each user account information source, such as an RDBMS database or a local text file. The Spring Security authentication process uses an object model that provides the following functionality:

- **Authentication manager**: The manager is responsible for creating and calling one of more authentication providers until the authentication process is complete (success or failure).

- **Authentication provider**: The provider is called by the authentication manager and is responsible for creating the user details service.

- **User details service**: This service is created by the authentication provider and is responsible for interacting with the account system to retrieve the user account along with the account attributes information, such as password and user roles. This information is used for validating the account password and for creating a Spring Security authentication token.

- **Authentication token**: This token is created as the result of a successful Spring Security authentication process. The token contains authenticated user information, such as user ID, granted roles, and session ID.

**Note**: The authentication token is used to generate an OpenEdge ClientPrincipal token that can be used further in the authorization process.

For details, see the following topics:

- Spring Security configuration files
• Enabling security for REST, WEB, APSV and SOAP
• User account configurations
• Login models
• CORS support
• Single sign-on (SSO) support in PAS for OpenEdge

Spring Security configuration files

In PAS for OpenEdge, you configure Spring Security by modifying the security properties file
(oeablSecurity.properties), and the URL access control file oeablSecurity.csv) instead of changing
bean constructor and property settings in multiple Spring Security XML files. These external properties files
are designed to be release independent, to be open to external management, and to provide access to all of
the security process functionality in the Spring Security framework.

Note: The Spring Security XML files are in the WEB-INF/spring directory on the server, but there is usually
no need to edit them directly.

Security properties files

The oeablSecurity.properties file can exist in three-tiered hierarchy so that you can:

1. Configure properties that are used for all ABL applications deployed on the instance by setting properties
   in instance-name/conf/oeablSecurity.properties. The defaults established in this file can be
   overridden by properties set in an ABL application (#2) or a web application (#3).

2. Configure properties that are used for ABL applications by setting properties in
   instance-name/ablapps/abl-app-name/oeablSecurity.properties. Property settings in this
   file override the server instance defaults (#1).

   Note: This level is optional. It is useful when you have more than one ABL application deployed on an
   instance and when those ABL applications require different security configurations.

3. Configure properties for a specific web application in an ABL application by setting properties in
   instance-name/webapps/web-app-name/WEB-INF/oeablSecurity.properties. Property settings
   in this file override the server instance defaults (#1) and the ABL application defaults (#2).

The Spring Security properties and their values are documented in the oeablSecurity.properties.README
file.

For some properties, the oeablSecurity.properties file employs a naming convention
(bean-name.property=value) that simplifies the identification of a bean and its properties. For example:

OECclientPrincipalFilter.sealAnonymous=false
Note:
Beginning with the OpenEdge 11.7 release, property=value pairs replace references to specific XML security model configuration files. For example, if you want to specify basic HTTP authentication using a local users file you would specify this set of properties in oeablSecurity.properties:

```plaintext
http.all.authmanager=local
client.login.model=basic
```

Prior to the OpenEdge 11.7 release, you would have to have to sort through a number XML templates to find one that matches the required security model, update it, and reference it from the web.xml file.

**URL access control file**

The instance-name/webapps/web-app-name/WEB-INF/oeablSecurity.csv file implements URL access controls for web applications. Each entry (or line) in the file is an ordered set of three values.

Note: Access control lists, since they are ordered sets of three values, do not lend themselves well to the format of a properties file with its name/value pairs. Therefore, URL access controls were not included in the oeablSecurity.properties file. CSV files are more suitable for access control lists, and they are easily maintainable by many external administrative tools.

The three values of an entry in oeablSecurity.csv correspond to the three attributes of a Spring Security intercept-url element, namely:

- **pattern** — the URL pattern which can include wildcards and regular expressions
- **method** — the HTTP access method
- **access** — role[s] that are allowed access to the resource

Each line in the file must contain all three values specified in a comma separated list.

For example, the following snippet grants access to any user who has either ROLE_PSCAdmin or ROLE_PSCUser privileges to data from a resource whose URL begins with /web/sales/.

```plaintext
"/web/sales/**", "GET", "hasAnyRole('ROLE_PSCAdmin','ROLE_PSCUser')"
```

**Enabling security for REST, WEB, APSV and SOAP**

Spring security is automatically enabled for validating clients of REST and WEB applications. You must update oeablSecurity.properties to enable Spring Security to validate clients of APSV and SOAP applications.

For APSV, set the apsv.security.enable property to basic. By default, the property is set to none.

For SOAP, set the soap.security.enable property to basic. By default, the property is set to none.

For more information, see the oeablSecurity.properties.README file.
User account configurations

You configure user account sources for web applications by setting the `http.all.authmanager` property in the `oeablSecurity.properties` file. The syntax is:

```
http.all.authmanager={local|extlocal|ldap|ad|oerealm}
```

For more information see the `oeablSecurity.properties.README` file.

See also

- Local and Extended Local user accounts on page 94
- LDAP user accounts on page 96
- OERealm user account configurations on page 97

Local and Extended Local user accounts

Both local and extended local configurations use an ABL application's `WEB-INF/users.properties` file as the source for user account information.

Local configurations

The local model is usually used during application development, where you would want a quick way of testing a web application's security structure. It allows you to easily edit accounts and add clear-text passwords in order to test user access based on roles.

To implement the local model, add the following setting in the Security and authentication models section of the `oeablSecurity.properties` file:

```
http.all.authmanager=local
```

See the `oeablSecurity.properties.README` file for more information.

After choosing the local security configuration model for a web application, you can add, remove, modify users or user authentication settings by updating the ABL application's `WEB-INF/users.properties` file.

Extended local configurations

Like the local model, the extended local model uses the `WEB-INF/users.properties` as the source for user account information. However, the extended local model is more secure because it allows you to use encrypted passwords. Encrypted (i.e. hashed/salted) passwords are generated with the GENSPRINGPWD utility. See Generating encrypted passwords with GENSPRINGPWD on page 95
To implement the extended local model, add the following setting in the Security and authentication models section of the oeablSecurity.properties file:

```
http.all.authmanager=extlocal
```

See the oeablSecurity.properties.README file for more information.

**The users.properties file**

The following shows the syntax of entries in users.properties:

```
username=password, role1, [role2, ..., roleN] { enabled | disabled }
```

The following shows the content of the default users.properties file:

```
restuser=password,ROLE_PSCUser,enabled
restdebug=password,ROLE_PSCUser,ROLE_PSCDebug,enabled
```

The contents of the users.properties file are included as an example and they are not secure since they are stored as plain text. Typically, for a production server, you would generate encrypted passwords, and add the secure passwords to the users.properties file. See Generating encrypted passwords with GENSPRINGPWD on page 95 for more information.

**Generating encrypted passwords with GENSPRINGPWD**

To provide greater security for user accounts stored in users.properties, you can use the GENSPRINGPWD command-line utility (oe-install-dir/bin/genspringpwd) to generate encrypted passwords for the entries in the file.

**GENSPRINGPWD Syntax**

```
genspringpwd password {bcrypt | sha256 | sha512 | sha1} [salt-value]
```

- **password**
  
  A character string representing the password to be encrypted.

- **bcrypt | sha256 | sha512 | sha1**
  
  Specify which hashing algorithm to use for encryption. The default is bcrypt.

- **salt-value**
  
  A string of 6 to 11 characters used to initialize the sha1, sha256, and sha512 algorithms with secret information known only to the creator and verifier of the hash value of the encrypted password. It cannot be used for the bcrypt algorithm,
Note: For more information on hashing and salt values, search for hash and salt cryptography references on the Web.

Using encrypted passwords
After you have created an encrypted password with GENSPRINGPWD, copy and paste the generated value to the password field in a user.properties entry, using the following syntax:

```plaintext
username=password, role1, [role2, ..., roleN] { enabled | disabled }
```

LDAP user accounts
The Lightweight Directory Access Protocol (LDAP) is a protocol for authenticating user credentials that are stored on an LDAP-compliant server. If you are not familiar with LDAP implementation and user account management, refer to LDAP administration documentation online.

To implement the local model, add the following setting in the Security and authentication models section of the oeablSecurity.properties file:

```plaintext
http.all.authmanager=ldap
```

You configure LDAP in the LDAP Authentication Manager section of the oeablSecurity.properties file. See oeablSecurity.properties.README for an explanation of LDAP properties and their valid settings.

Note: The LDAP UserDetails implementation in Spring Security automatically updates role names. Ensure that role names are upper case since searches are case sensitive.

After configuring the LDAP model for your Web application, you can implement CORS support.

Active Directory user accounts
MicroSoft's Active Directory (AD) service provides a namespace for resolving wide range of objects including user accounts. An Active Directory configuration for web applications deployed on PAS for OpenEdge allows authentication using a Windows user@domain type ID. It locates and validates users from the user object's userPrincipalName field. It obtains granted roles from the user object's memberOf field.

To implement the Active Directory security model, add the following setting in the Security and authentication models section of the oeablSecurity.properties file:

```plaintext
http.all.authmanager=ad
```

You configure AD in the Microsoft Active Directory user account services section of the oeablSecurity.properties file. See oeablSecurity.properties.README for an explanation of AD properties and their valid settings.
Note: If more configuration detail is needed, use the full LDAP configuration described in the LDAP Authentication Manager section of the oeablSecurity.properties.README file.

OERealm user account configurations

The OERealm model allows you to use an OpenEdge application server as a source of user account information for the authentication process.

To implement the OERealm security model, add the following setting in the OERealm user account server configuration section of the oeablSecurity.properties file:

```
http.all.authmanager=oerealm
```

You configure it in the OERealm user account server configuration section of the oeablSecurity.properties file. See oeablSecurity.properties.README for more information.

Note that an OERealm implementation consists of the following components:

• **OpenEdge Realm (OERealm) Client:** The client contains the Spring Security authentication provider and user account service objects that interact with the application server to retrieve the user account information. This information is then sent to the Spring Security authentication process.

  The OERealm Client does the following:

  1. Connects to the application server.
  2. Executes a request to locate the user account and retrieve the account unique numeric ID.
  3. Executes requests to retrieve the user account details in the following order:
     a. Granted roles
     b. The account enabled state
     c. The account locked state
     d. The account expired state

  Note: If the OERealm client finds that the account does not exist or if the account is in the disabled, locked, or expired state, no additional AppServer requests are performed.

• **OERealm AppServer ABL interface:** An ABL class that runs on the AppServer and responds to requests from the OERealm client and implements the interface IHybridRealm.

  The actions of the class is performed in the following order:

  1. Search for the user account using the name (userid[@domain]).
  2. Return the user account attributes for the user account.
  3. Validate the user account’s password.

     Optionally, authorize the OERealm client and enable it to receive user account information.

  For more information
Creating an OERealm service interface

To implement OpenEdge SPA, you must create an ABL class that implements the OERealm service interface that finds user account information, obtains user attributes, and validates a user account's password.

To configure an OERealm service:

1. Configure a OpenEdge Application Server to host the OERealm service interface class.

   **Note:** Either a PAS for OpenEdge or an OpenEdge AppServer instance can host the OERealm service interface class. If you are using the OpenEdge AppServer, it must be configured as a STATE-FREE instance. (A PAS for OpenEdge instance functions in any operating mode.)

2. Create a singleton OOABL class that implements the OERealm HybridRealm interface. If you have the OpenEdge Documentation and Samples installed, you can find a sample interface implementation class, HybridRealm.cls, at $DLC\src\samples\security\OpenEdge\security\Realm.

   In the sample HybridRealm.cls file, the following methods are declared and defined:

   **Table 10: Methods defined in the HybridRealm.cls class**

<table>
<thead>
<tr>
<th>Method</th>
<th>Mandatory?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetAttribute</td>
<td>Yes</td>
<td>This method retrieves an attribute for the specified user account.</td>
</tr>
<tr>
<td>GetAttributeNames</td>
<td>No</td>
<td>This method retrieves the name assigned to the attributes.</td>
</tr>
<tr>
<td>GetUserNames</td>
<td>No</td>
<td>This method retrieves all the user names associated with the user account.</td>
</tr>
<tr>
<td>GetUserNamesByQuery</td>
<td>No</td>
<td>This method retrieves the user names that match the input query string.</td>
</tr>
<tr>
<td>RemoveAttribute</td>
<td>No</td>
<td>This method deletes the value of the given attribute.</td>
</tr>
<tr>
<td>SetAttribute</td>
<td>No</td>
<td>This method assigns a new value to the given attribute.</td>
</tr>
</tbody>
</table>
## User account configurations

<table>
<thead>
<tr>
<th>Method</th>
<th>Mandatory?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ValidatePassword</td>
<td>Yes</td>
<td>This method validates the client-supplied password of the user account against the stored password.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong>: You can define both the clear-text and digest form of the ValidatePassword() method.</td>
</tr>
<tr>
<td>ValidateUser</td>
<td>Yes</td>
<td>This method searches for the user account, validates whether the account domain is valid and enabled, and returns the account numeric ID.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong>: The OERealm service interface uses a numerical user account ID as an ABL integer instead of the full user account name. If the user account storage of your OERealm service does not support the integer user account IDs, it is recommended that you build a mechanism to permanently assign unique ABL integer values to each user account in the storage.</td>
</tr>
</tbody>
</table>

You can implement the `GetAttribute()` method to return the `UNKNOWN` or `CHARACTER` value. If you have configured the Web application with an SPA security configuration model, the `OERealmUserDetailsImpl` module of the OERealm client handles the `UNKNOWN` value by replacing the value with a static, non-configurable default value. If a `CHARACTER` value is returned, the OERealm service interface must encode the value in a format that the `OERealmUserDetailsImpl` module can understand.

The known values of the `OERealmUserDetailsImpl` module are:

### Table 11: OERealmUserDetailsImpl module attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Data type</th>
<th>Character encoding format</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTR_ROLES</td>
<td>string-extent</td>
<td>&quot;&quot; or &quot;xxxx[,,yyyy,zzzz]&quot;</td>
<td>The OERealm service interface must return roles without the ROLE_ prefix. For example, the interface must return PSCUser instead of ROLE_PSCUser. These roles are used in oeablSecurity-xxxxx-oerealm.xml grant/deny access to URL/method combinations and will be used by the external security as part of its authorization functions.</td>
</tr>
<tr>
<td>ATTR_ENABLED</td>
<td>boolean</td>
<td>A character from the {T,t,F,f,Y,y,N,n,0,1} set.</td>
<td>T,t,Y,y,1 denotes that the user is enabled. F,f,N,n,0 denotes that the user is disabled.</td>
</tr>
</tbody>
</table>
### OERealm security considerations

The OERealm authentication process must be secure against unwarranted attacks.

The following considerations must be observed for security purposes:

- **OERealm service interface security**

  If the ABL class that provides the OERealm service requires authentication of clients, then the client must have a realm token property configured with a file that contains a sealed ClientPrincipal that the OERealm service is capable of validating.

  Since the OERealm user details service can be configured to send a sealed ClientPrincipal to the AppServer OERealm service interface on each method it invokes, the OERealm service interface can wrongly intercept the ClientPrincipal from `SESSION:CURRENT-REQUEST-INFO:GET-CLIENT-PRINCIPAL` and validate the client before performing any OERealm operation.

  Another security flaw occurs when you use the `SECURITY-POLICY:SET-CLIENT()` or `SET-DB-CLIENT()` method for performing OERealm client identity checks. Using these methods changes the current identity of the ABL session, which must then be reestablished after the OERealm operation is complete.

  To overcome this problem, it is recommended that you use the `CLIENT-PRINCIPAL:VALIDATE-SEAL()` method along with encoded domain access codes ("oech1:xxxx") to validate the ClientPrincipal. You can apply this validation as an operation at the beginning of each OERealm service interface method. This isolates the OERealm use case from the other ClientPrincipal-related activities, such as a session, database, AppServer connection, or BPM identity.

- **OERealmAuthProvider security**

  You can configure the OpenEdge domain access code to seal the ClientPrincipal in the Single sign-on (SSO) mode. By default, the Spring Security `OERealmAuthProvider` bean creates the ClientPrincipal as a single-tenant. The problem is that every Web application user of the ClientPrincipal is a member of the same OpenEdge domain.

  You can enhance `OERealmAuthProvider` security in one of the following ways:

    - Set the `userDomain` and `key` properties.
You can enhance security by setting the following OERealmAuthProvider properties in the oeablSecurity.properties file while sealing the ClientPrincipal:

- The OERealmAuthProvider.userDomain attribute to set the static OpenEdge domain.
- The OEClientPrincipalFilter.key attribute to set the domain access code.

These attributes are used to uniquely identify application users based on the user domain. You can also disable access to the Web application just by disabling the domain. The following table defines the available combinations of domain, domain access code, and user ID:

Table 12: Available combinations of domain, domain access code, and user ID

<table>
<thead>
<tr>
<th>Domain name</th>
<th>Key attribute</th>
<th>User ID</th>
<th>Signing operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;&quot; (default)</td>
<td>&quot;&quot; (default)</td>
<td>&quot;uid&quot; (&quot;uid@&quot;)</td>
<td>Seal the blank domain using the built-in Domain Registry blank access code.</td>
</tr>
<tr>
<td>&quot;&quot;</td>
<td>&quot;&lt;xxxxx&gt;&quot;</td>
<td>&quot;uid&quot; (&quot;uid@&quot;)</td>
<td>Seal the blank domain using the clear text sting value, &quot;&lt;xxxxx&gt;&quot;.</td>
</tr>
<tr>
<td>&quot;&quot;</td>
<td>&quot;oechl:&lt;xxxxx&gt;&quot;</td>
<td>&quot;uid&quot; (&quot;uid@&quot;)</td>
<td>Seal the blank domain using the encoded value, &lt;xxxxx&gt;.</td>
</tr>
<tr>
<td>&quot;abc&quot;</td>
<td>&quot;&quot; (default)</td>
<td>&quot;uid@abc&quot;</td>
<td>Seal the abc domain using the built-in Domain Registry blank access code.</td>
</tr>
<tr>
<td>&quot;abc&quot;</td>
<td>&quot;&lt;xxxxx&gt;&quot;</td>
<td>&quot;uid@abc&quot;</td>
<td>Seal the abc domain using the clear text sting value, &quot;&lt;xxxxx&gt;&quot;.</td>
</tr>
<tr>
<td>&quot;abc&quot;</td>
<td>&quot;oechl:&lt;xxxxx&gt;&quot;</td>
<td>&quot;uid@abc&quot;</td>
<td>Seal the abc domain using the encoded value, &lt;xxxxx&gt;.</td>
</tr>
</tbody>
</table>

- Set the multiTenant property.

If you enable the OERealmAuthProvider.multiTenant property, all the user IDs are appended with the userDomain value only if the user-id being authenticated does not already specify a Domain name.

- Set the External domain registry for multi-tenant applications.

The OERealmAuthProvider provides support for using an administrator-generated Domain Registry file to sign and seal ClientPrincipals in multi-tenant applications. This file contains a list of domains that are allowed to have access to the ClientPrincipal.

To use the Domain Registry file:

1. Use the gendomreg command, located at $DLC/bin, that takes a formatted text file as input and creates a secure binary Domain Registry file. The text file lists OE domains that are allowed to have access to the ClientPrincipal in the Domain-name=Domain-access-code format. Each domain must be listed in a separate line.

2. Copy the Domain Registry file to the OE REST Web application’s /WEB-INF/classes directory.

3. Configure the OERealmAuthProvider bean to load and sign a ClientPrincipal.

- Preventing authorization before authentication
The ClientPrincipal that is generated by the Web application's OERealmAuthProvider supersedes the one that is generated by the OEClientPrincipalFilter for SSO. This is a problem when the OEClientPrincipalFilter authorizes the incorrect ClientPrincipal without any processing. You can disable the OEClientPrincipalFilter by setting the OEClientPrincipalFilter.enabled property to false.

Another potential security problem occurs if a client is able to call the OERealm service interface without authentication. You can secure the service interface against such unwarranted client access by using the ClientPrincipal object and validation technique that is similar to the one that is used in the ABL business logic of your OpenEdge application server. To use this validation technique, the Web application's OERealmUserDetailsImp module acts as a client and identifies itself to the application server's OERealm service interface. The service interface validates the client using a ClientPrincipal just like it is used in other parts of the ABL application.

To develop the Web application's OERealmUserDetails for acting as an OpenEdge application server client:

1. Use the genspacp utility, at the $DLC/bin location, to create a sealed ClientPrincipal that represents a OERealm service client. The utility requires the user ID, domain name, domain access code, and output file name to create a binary file that contains the sealed ClientPrincipal in the <file-name>.cp format. The domain name and domain access code correspond to those that are then used in the OERealm service interface to validate clients.

2. Copy the output file from the $DLC/bin location to the Web application's WEB-INF/classes directory.

3. Set the OERealm.UserDetails.realmTokenFile property to the output file name.

4. In the OERealm service interface, add the ClientPrincipal validation code. The code can reside in the AppServer activate procedure or as step in each of the OERealm class methods. It is recommended that you use the CLIENT-PRINCIPAL:VALIDATE-SEAL("domain-access-code") method for validation. You must ensure that domain-access-code is in the encoded "oecli::xxxx" format.

When the OERealm service interface validates the ClientPrincipal in the created binary output file, the following sequence occurs:

1. Once the Web application is loaded, if the OERealm.UserDetails.realmTokenFile property contains the output file name, the file is used to load the ClientPrincipal. If the output file does not load, an error is added to the error log.

2. For each user authentication request, OERealmUserDetails is used to send the ClientPrincipal to the AppServer OERealm service interface methods.

3. The OERealm service interface validates the ClientPrincipal. The following validation outcomes are possible:
   - If the ClientPrincipal is not sent to the interface, an error is returned.
   - If the ClientPrincipal is sent to the interface, the interface is used to validate the domain access code. If the validation fails, an error is returned.
   - If the ClientPrincipal validation is successful, the execution of the ABL class method continues.

Multi-domain support for user accounts

This topic describes how PAS for OpenEdge extends multi-domain support for user accounts.

Simple multi-domain support in PAS for OpenEdge accommodates implementations where the user-id entered by the client is a fully qualified OpenEdge ID. Fully qualified means that both the user-id and the domain (or tenant) name are included. This type of user authentication requires the back-end user account storage to be implemented with multiple user domain capabilities.
An OpenEdge domain is a group of user accounts that all share the same access control rights. A tenancy is a security feature that attributes access control to data for all user accounts of all the domains associated with a single tenant. A tenant is required to have one domain in which all user account members are members of the tenant. Any tenant may be associated with multiple domains.

In addition, PAS for OpenEdge includes multi-domain support for implementations where the back-end system’s user account storage does not specify domains or tenancy. The primary example of this is an LDAP authentication provider. The user account storage on an LDAP server is a flat space holding all user accounts for all domains, and each account has a unique ID. In this case, the domain name is not something the client supplies during authentication, but rather something that must be related to the user account by configuring the OEClientPrincipalFilter bean in the OEClientPrincipalFilter bean section of the oeablSecurity.properties file. It has capabilities that allow it to derive a domain name from a user account’s granted roles.

This method of using Spring Security roles as OpenEdge domains follows the logical pathway where a user account becomes a member of an OpenEdge domain by virtue of being granted membership in the domain. This is the same reasoning used in LDAP directories when an administrator associates individual user accounts with an LDAP group for the purpose of controlling access.

For more information on multi-domain support for authentication systems like LDAP see the following topics:

- Resolving user-id and domain names
- Obtaining a domain name from a role name
- Configuring multiple domains
- OpenEdge domains in LDAP configurations

Resolving user-id and domain names

The OEClientPrincipalFilter contains the logic that processes an OpenEdge ID to yield a user-id and domain name. The user-id and domain are used in creating a client-principal object that is subsequently passed to the ABL application code.

The OEClientPrincipalFilter logic follows these general steps after an authentication provider has successfully authenticated the user:

1. If the input OpenEdge ID is fully qualified (user-id plus domain name), proceed to Step 3.
2. Try to resolve the domain name by doing the following:
   a. First, look for a user role in the OEClientPrincipalFilter.domainRoleFilter property, which is recognized as holding a domain name. If the domain name is found, proceed to Step 3.
   b. Next, try using the OEClientPrincipalFilter.domain property value. If the domain name is found, proceed to Step 3.
   c. If the domain name remains unresolved after Steps 2a and 2b, raise an error condition.
3. Create a client-principal object and populate its data fields.
4. If the OEClientPrincipalFilter.key property is configured, use its value as the Domain Access Code (DAC) for all domain names.

Note: All OEClientPrincipalFilter properties are set in the oeablSecurity.properties file.
Note: The OEClientPrincipalFilter.key and the OEClientPrincipalFilter.registryFile properties are mutually exclusive. Both cannot be configured at the same time.

5. If the OEClientPrincipalFilter.registryFile property is configured, use the domain name (either input directly in Step 1 or resolved by one of the methods in Step 2) to look up the domain’s DAC.

6. If the DAC cannot be resolved, raise an error condition.

7. Seal the client-principal with the DAC and store it in the user’s HTTP login session context for use in subsequent client requests.

Obtaining a domain name from a role name

The OEClientPrincipalFilter.domainRoleFilter property of the Spring Security OEClientPrincipalFilter bean obtains an OpenEdge domain name from a user-granted role. The domainRoleFilter property is a Java RegEx (regular expression) pattern that is used to identify role names produced by Spring Security and to extract the domain name from them. The basic requirement is to adopt a role-naming convention that distinguishes role names that represent domains from other role names.

The Java RegEx expression can be any valid pattern with a single extraction group (a Java RegEx pattern enclosed in parentheses). The full pattern is used to match a role name. Anything that matches the pattern enclosed in parentheses is extracted and becomes a domain name.

The following are examples of user role-naming conventions that define domains:

<table>
<thead>
<tr>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OED:(.*)</td>
<td>The domain name is everything after OED:</td>
</tr>
<tr>
<td>Domain-(.*)-name</td>
<td>The domain name is all characters between Domain- and -name</td>
</tr>
</tbody>
</table>

Spring Security authentication providers (the beans that authenticate user accounts) prefix ROLE_ to all role names provided for authenticated users. Since the role names presented to the OEClientPrincipalFilter bean are in the form ROLE_user-role-name, this is what would be matched by the domainRoleFilter property:

- ROLE_OED:(.*)
- ROLE_Domain-(.*)-name

When the domainRoleFilter property is not defined or is a blank string, looking for a domain name in the user’s list of granted roles is disabled.

Configuring multiple domains

Configuring for multi-domain support begins with the following steps:

1. Design the OpenEdge domain name space to be used by the ABL application, and assign a Domain Access Code (DAC) to each domain name.

2. Build a CSV text file with domain names and clear-text DAC pairs, one pair per line. Domain names and DACs are separated by a comma. For example:

```
domain-name, clear-text-DAC
```
3. Use the gendomreg utility to create an encrypted Java keystore file from the CSV file. For example:

```
oe-install-dir/bin/gendomreg domreg.csv domreg.keystore
```

4. Copy the encrypted Java keystore file to each PAS for OpenEdge instance's `.../conf` directory.

5. Configure the OEClientPrincipalFilter bean in the `oeablSecurity.properties` file as follows:
   a. Set the `OEClientPrincipalFilter.key` property to a blank value.
   b. Set the `OEClientPrincipalFilter.registryFile` property and set its value to the file name of the Java keystore that you created in Step 3. For example:

```
OEClientPrincipalFilter.registryFile=domreg.keystore
```
   c. Set the `OEClientPrincipalFilter.domain` property to a default domain that has minimal access rights to your ABL application.
   d. Optionally configure `authz`, `acctinfo`, `roles`, and other properties.

The next step is to design the role name pattern that will be used to designate an OpenEdge domain. Ideally the name pattern will have a unique prefix that distinguishes it from all other names. The name pattern may also employ a postfix to help establish uniqueness. See Obtaining a domain name from a role name on page 104 for more information about designing a role name pattern.

**Note:** Since the OEClientPrincipalFilter bean uses Java RegEx functionality, create a role name pattern that is easy for Java RegEx to parse.

When you have established a role name pattern, construct the Java RegEx pattern that will distinguish the domain name from any other normal user account granted roles. For example, with `OED:domain-name` as the domain name:

<table>
<thead>
<tr>
<th>Role Name</th>
<th>Spring Security Role Name</th>
<th>Java RegEx Match Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROLE_OED:acme</td>
<td>ROLE_OED:acme</td>
<td>ROLE_OED:(.*)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Role Name</th>
<th>Spring Security Role Name</th>
<th>Java RegEx Match Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>cn=OED:acme</td>
<td>ROLE_OED:ACME</td>
<td>ROLE_OED:(.*)</td>
</tr>
</tbody>
</table>

The Spring LDAP Authentication Provider prefixes `ROLE_` to any LDAP group name it uses as a user account role, and it converts the LDAP group name to upper case.

Finally, set the OEClientPrincipalFilter bean's domainRoleFilter property to the value of the Java RegEx match pattern.
Note: If your Java RegEx match pattern contains a back slash (\), you may be required to escape it with another backslash (\\).

OpenEdge domains in LDAP configurations

Spring Security LDAP configuration details will almost always be unique to each individual enterprise; however, the following will give you information about configuring OpenEdge domains in the directory service so that the OEClientPrincipalFilter bean can find and use them.

Each LDAP directory service has a schema that defines its objects and object attributes. Different implementations of a directory service may vary in the objects and object attributes it supports. Examples of different directory services are:

- Apache Directory Service
- Windows Active Directory
- OpenLDAP

An OpenEdge domain will be just another LDAP group object that contains a list of the user accounts (Distinguished Names) who are members of that group, and who are entitled to the access associated with the group. The difference is that the LDAP group's name will be based on the name pattern you defined in the previous sections.

For example, directory service administrators use the LDAP groupofUniqueNames objects as user account roles, where each group member's user account is in a uniqueMember attribute. The directory service's hierarchical object space might look like the following:

```
dc=acme
   ou=users
      cn=acmeuser1,ou=users,dc=acme,dc=com
   ou=groups
      cn=OED:acme,ou=groups,dc=acme,dc=com
      uniqueMember=cn=acmeuser1,ou=users,dc=acme,dc=com
```

In this simple example, the user account `acmeuser1` is in the domain `acme`. This occurs because a group is defined with the name `OED:acme`, and the user account for `acmeuser1` is a member of that group. Therefore, when Spring Security authenticates `acmeuser1`, it will see that `acmeuser1` is a member in the `ROLE_OED:acme`. The role supplies the domain name `acme` for creating a client-principal's domain-name field.

For more information about LDAP, see LDAP user accounts on page 96

Login models

PAS for OpenEdge supports the following login models for web applications:

- **basic** — typically used by client authentication to stateless ABL business application architectures. There are no configurable properties.
- **anonymous** — used as the default for getting started with the development of an ABL web application. It allows unauthenticated client access to the ABL web application until it is mature enough to use product quality authentication. There are no configurable properties.
• container — used when the ABL web application wants to use the Realm authentication provided by the Apache Tomcat web server it runs in. This can be useful when synchronizing user accounts with other 3rd party Java web applications in the same PAS for OpenEdge instance. Set the http.jee.all.mappableRoles property to a comma separated list of role names from the Tomcat realm's authentication.

• form — typically used for managing the login/logout of clients to a stateful ABL business application architecture. Properties are found in the Form login Filter bean and Logout Filter bean section of the oeablSecurity.properties file.

• sso — login via a ClientPrincipal token that represents a user. SSO properties are found in the OpenEdge SSO support section of oeablSecurity.properties. Also see Single sign-on (SSO) support in PAS for OpenEdge on page 108.

You set the login model as a value of the client.login.model property. See oeablSecurity.properties.README for more information.

CORS support

Cross-origin resource sharing (CORS) is a W3C group standard that allows an HTTP client to access a Web page from one domain, and to access resources located in another domain. Such “cross-domain” requests are otherwise forbidden by a Web browser's default, same-origin security policy. The CORS standard defines a way in which a client can ask a Web server if it can access the cross-origin request. The Web server's configuration determines if the cross-domain request will be granted.

The CORS standard was designed primarily to support JavaScript clients, however it may be applicable to other clients as well.

The CORS standard is implemented through HTTP headers that allow resource access to permitted domains. Most modern browsers support these headers and enforce the restrictions these headers establish. Additionally, for HTTP request methods that might affect user data (in particular, methods other than GET or POST with certain MIME types), the specification mandates that the browser preflight the request. Preflighting is a check to determine if the HTTP request is safe to send to the other domain. The server solicits supported methods from the other domain with an HTTP OPTIONS request header. Upon approval of the preflight request, the actual HTTP request is sent. Servers can also notify clients whether user credentials (including cookies and HTTP authentication data) can be sent with requests.

Note:

Although CORS support is extended to virtually all modern browsers, check to ensure that your browser supports the CORS standard.

Also note that you must manage security constraints (roles, for example) in the Web application's oeablSecurity.properties file. See the OECORSFilter bean of the oeablSecurity.properties.README for more information. Also refer to the Web servlet standard documentation online for more information.

A CORS enabled server or Web application classifies all HTTP requests as:

• A CORS request that contains the HTTP Origin header

• A preflight request that contains the Access-Control-Request-Method header in an OPTIONS request

• A generic request that does not contain any CORS HTTP headers
OpenEdge uses a third-party Java open source package, CORS Filter, in which all of the primary CORS functionality resides. OpenEdge has integrated CORS filter into the Java container Web applications by implementing a Spring Security filter bean so that the CORS filter can be configured from within the Spring Security configuration files, with all of the other Web application security.

Therefore, the CORS support is already incorporated into the Web application security policy, and is enabled for all generic requests and CORS requests from any domain.

For more information on CORS standard and the advances in the standard, see the documentation at http://www.w3.org/TR/cors/.

**Note:** CORS can be used as an alternative to the JSON with padding (JSONP) pattern. While JSONP supports only the GET request method, CORS also supports other types of HTTP requests. Using CORS enables a Web programmer to use the XMLHttpRequest API, which supports better error handling than JSONP. Also, while JSONP can cause XSS issues where the external site is compromised, CORS allows Web sites to manually parse responses to ensure security.

### Adding CORS support

To add CORS support:

1. Open the `WEB-INF/oeablSecurity.properties` file of your web application.
2. Scroll down to the `OECORSFilter` bean section.
3. Set OECORSFilter properties.
   
   See the `WEB-INF/oeablSecurity.properties.README` for information about OECORSFilter properties.

4. After saving your changes, restart the PAS for OpenEdge instance.

### Single sign-on (SSO) support in PAS for OpenEdge

Single sign-on (SSO) in OpenEdge refers to the creation of a sealed security token called the ClientPrincipal that represents a user identity within a security domain. The ClientPrincipal is created after Spring Security successfully completes authentication and authorization. The ClientPrincipal retains the user's credentials and remains available within a Web application, eliminating the need for a user to login again.

In PAS for OpenEdge, basic Spring Security token generation is extended to create an OpenEdge ClientPrincipal object. Creation of the ClientPrincipal is enabled by default by this property in the web application's `oeablSecurity.properties` file:

```properties
OEClientPrincipalFilter.enabled=true
```

Set to `false` to disable the ClientPrincipal. You can set other ClientPrincipal properties in the `OEClientPrincipalFilter` bean section of the `oeablSecurity.properties` file. In particular, you will want to set the `OEClientPrincipalFilter.domain` property to specify where the ClientPrincipal is valid.

After domain and other properties are set for the ClientPrincipal, you can configure SSO properties in the `OpenEdge SSO support` section of the `oeablSecurity.properties` file:

See the `oeablSecurity.properties.README` file for more information on the properties and values for the ClientPrincipal and for SSO.
Note: In PAS for OpenEdge, SSO is available for client access via the APSV and REST transports but not for the SOAP transport.
WebSpeed configuration and management

This section contains configuration and management information specific to WebSpeed support on Progress Application Server for OpenEdge.

**Note:** Also see Modifying the WEB transport URL on page 21 for information about changing the default transport that handles WebSpeed requests.

Beginning with OpenEdge 11.6, PAS for OpenEdge supports deployment of WebSpeed applications. An instance of Progress Application Server for OpenEdge combines the functionality of a Web server and an application server. See *Introducing PAS for OpenEdge* for an overview of the architecture and advantages of WebSpeed on PAS for OpenEdge.

For details, see the following topics:

- Tools for managing and configuring WebSpeed on PAS for OpenEdge
- Configuring openedge.properties
- Security
- Migrating classic WebSpeed applications

Tools for managing and configuring WebSpeed on PAS for OpenEdge

This section is an overview of the tools for configuring and managing WebSpeed on PAS for OpenEdge.
Note: WebTools, a collection of browser-based utilities implemented for classic WebSpeed, are not supported for use with WebSpeed on PAS for OpenEdge.

Command line utilities
The command line utilities for classic WebSpeed (WTBMAN and WSCONFIG) are not applicable to WebSpeed on PAS for OpenEdge. Instead, you use the standard PAS for OpenEdge utilities, namely:

- TCMAN — Similar to WTBMAN, it is used to control and monitor the server. TCMAN has additional functionality, compared to WTBMAN, for creating and configuring server instances, managing applications, and more.

- OEPROP — Similar to MERGEPROP, which is used to manage the ubroker.properties file for classic WebSpeed. You use OEPROP to manage the openedge.properties file, which is similar in form and function to ubroker.properties.

REST APIs
The oemanager.war Java Web application provides a REST API for remote management and monitoring of sessions, agents, transports, and of ABL applications deployed on a Progress Application Server for OpenEdge instance. Functionality is implemented as a REST service that is accessed via a URI.

OEE/OEM
You can configure and manage both classic WebSpeed and WebSpeed on PAS for OpenEdge from the OpenEdge Explorer or OpenEdge Management tools. For more information, see the online help for OEE/OEM. For information specific to managing WebSpeed on PAS for OpenEdge, see OpenEdge Management: Progress Application Server for OpenEdge Configuration.

Manually editing configuration files
You can edit openedge.properties (and other PAS properties files) with a text editor to configure. The properties are documented in the /conf/openedge.properties.README file.

Note: Be aware that manual edits cannot be tracked. Therefore, if manual changes are made, and changes are also made with TCMAN, OEPROP, REST APIs, or OEE/OEM tools, the manual changes may be lost.

See also
TCMAN Reference on page 115
OEPROP on page 45
REST API Reference for oemanager.war on page 159

Configuring openedge.properties
In classic WebSpeed, you configure the WebSpeed Transaction Server and define features with environment variables and entries in the ubroker.properties file. In PAS for OpenEdge, the $CATALINA_BASE/conf/openedge.properties file is used instead of ubroker.properties largely replaces environment variables. It is similar in format to ubroker.properties but applies only to a single server instance. That single instance, however, can support a number of connection and session types.
For example, you specify database connections for all deployed Web applications (including WebSpeed) in `openedge.properties` by setting the `agentStartupParam` parameter with the following syntax:

```
agentStartupParam=-db dbname [ -db dbname ... ]
```

In addition, `openedge.properties` contains a `ROOT.WEB` section the applies specifically to WebSpeed support. For example:

```
[oepas1.ROOT.WEB]
adapterEnabled=1
defaultCookieDomain=
defaultCookiePath=
defaultHandler=OpenEdge.Web.CompatibilityHandler
srvrDebug=0
```

You can find help that explains these parameters and their settings in `$CATALINA_BASE/conf/openedge.properties.README`

### Security

PASOE web transport’s default security is set very coarse grained, meaning that if a client is authenticated and has the right role they can access anything in the `/web/**` relative URI.

There are two possible solutions to adding additional fine grained access control:

- Write ABL code in your web handlers that authorizes the user (`Client-Principal`) to a particular handler’s URI or to sub paths
- Add Spring Security `<intercept-url>` elements in your `oeablSecurity-*.xml` configuration - with one `<intercept-url>` per web handler

The former exposes the ABL business logic to additional risk and requires yet another location where role based authorization has to be configured and implemented. The later is safer (because it runs in protected code space) and is easily configurable at the end production site.

### 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:

```
.
.
[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`. 
TCMAN Reference

TCMAN is a command-line utility for managing and administering the Progress Application Server. TCMAN extends the basic Tomcat scripts for starting, stopping, and managing server instances.

This TCMAN Reference contains usage information for the tcman command as well as syntax information on all of the TCMAN actions.

For details, see the following topics:

• Using TCMAN
• Manager actions
• Server actions
• General actions

Using TCMAN

You use the TCMAN utility by running the tcman command, with specified actions and options, from either the core PAS ($CATALINA_HOME/bin) or from an instance ($CATALINA_BASE/bin). You can run tcman from a command line, or in a script. Both Windows (tcman.bat) and UNIX (tcman.sh) versions of TCMAN are implemented in the Progress Application Server.

See also

The tcman command on page 116
Extending TCMAN on page 118
The tcman command

Purpose

TCMAN is a command-line utility for managing and administering PAS. On UNIX systems, you run the tcman.sh script followed by appropriate TCMAN actions and options. On Windows systems, you run the tcman.bat batch file, which is identical syntactically and functionally with tcman.sh.

Note: For the sake of brevity, all the syntax statements and examples in this reference show the tcman.sh script.

Syntax

```
{ $CATALINA_HOME | $CATALINA_BASE }/bin/tcman.sh action [general_options]
[ action_options ]
```

Parameters

$CATALINA_HOME|$CATALINA_BASE

Specify whether to run TCMAN from the root directory of the installed PAS ($CATALINA_HOME) or from the root directory of an instance ($CATALINA_BASE). The context of where you run TCMAN (whether from the /bin directory of the parent, or the /bin directory of an instance) affects which server the utility acts on.

Note: TCMAN automatically determines the value of CATALINA_BASE from the directory where you start it. When you run it from the /bin directory of an instance, the value of CATALINA_BASE is the root directory of the instance. If you run it from the /bin directory of the installed Progress Application Server, the value of CATALINA_BASE is the root directory of the installed server (which is the same value as CATALINA_HOME).

action

Specify which TCMAN action to invoke.

general_options

Specify one or more of the TCMAN common options that can apply to most actions. Note that one or more of the general options may be required by a specific action. For example, the list action requires –u in order to pass a user name and password.

The output of tcman.sh help action includes a list of general options that are applicable to a particular action.

The following table is a list of the common options:
Table 15: TCMAN general options

<table>
<thead>
<tr>
<th>Common options</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-u user_name:password</code></td>
<td>Pass a valid user name and a password for HTTP Basic access authentication.</td>
</tr>
<tr>
<td><code>-v</code></td>
<td>Display verbose output.</td>
</tr>
<tr>
<td><code>-M URL</code></td>
<td>Override the default manager that manages Web applications by specifying the URL of an alternative manager. URL is expressed in the following format: `{http</td>
</tr>
<tr>
<td><code>-B</code></td>
<td>Override default CATALINA_BASE environment settings.</td>
</tr>
<tr>
<td><code>-n</code></td>
<td>Debug the TCMAN action but do not execute changes.</td>
</tr>
<tr>
<td><code>-I instance_name</code></td>
<td>Run TCMAN from the /bin directory of the specified instance.</td>
</tr>
</tbody>
</table>

action_options

Specify an option that applies to the selected action. These options are explained in the topics that describe each action.
Example

Run the `help` action from the core server (/psc/pashome) to display a list of available TCMAN actions:

```
/psc/pashome/bin/tcman.sh help
usage: tcman action [options...]
manager actions:
  list       list deployed applications
  info       list server info
  deploy     deploy application
  undeploy   undeploy application
  reload     reload application
  status     show server status
  leaks      show server memory leaks
  enable     start web application running
  disable    stop running web application
  resources  list server global resources
  sessions   list a web application's sessions
server actions:
  create     create a new server instance
  delete     delete server instance
  config     dump CATALINA_BASE configuration
  clean      clean/archive log files
  instances  list tracked server instances
  register   manually register an instance
  unregister  manually unregister an instance
  start      start this server
  stop       stop this server
  version    show the server version information
  test       test the server's configuration
general actions:
  env        show tcman execution environment
  help       show this information
```

See also

Manager actions on page 120
Server actions on page 132
Extending TCMAN on page 118
General actions on page 155

Extending TCMAN

You can extend TCMAN by adding your own actions to the built-in actions of the TCMAN utility. You implement actions as UNIX shell scripts, Windows batch files, or Windows PowerShell scripts. Customizing TCMAN with your own actions allows you to unify all of your administrative tasks under a single utility.

File name format

Name your TCMAN action scripts using the following convention:

```
action_nameCmd{.sh | .bat | .ps1}
```
where file name extensions .sh, .bat, and .ps1 are used for UNIX scripts, Windows batch files, and PowerShell scripts respectively.

**UNIX action scripts**
UNIX action scripts must be executable. That is, the file must have execution permissions for the current user. Each script must exit with a zero (0) status when successful, or a positive non-zero value to indicate an error.

When the action script is in the core server's bin directory ($CATALINA_HOME/bin), it is available to administrators of any instance. When the action script is in an instance's bin directory ($CATALINA_BASE/bin), it is only available to administrators of that instance.

The search order is:
1. $CATALINA_BASE/bin
2. $CATALINA_HOME/bin

**Windows batch and PowerShell action scripts**
On Windows, action scripts can be either DOS batch scripts (.bat) or PowerShell scripts (.ps1). Batch scripts should return zero (0) for success, or a positive non-zero integer to indicate an error level. Powershell scripts should return \$true for success or \$false to indicate an error.

When the action script is in the core server's bin directory ($CATALINA_HOME/bin), it is available to administrators of any instance. When the action script is in an instance's bin directory ($CATALINA_BASE/bin), it is only available to administrators of that instance.

If you have both batch and PowerShell scripts, TCMAN gives precedence to batch files. The search order is:
1. $CATALINA_BASE/bin/action_nameCmd.bat
2. $CATALINA_BASE/bin/action_nameCmd.ps1
3. $CATALINA_HOME/bin/action_nameCmd.bat
4. $CATALINA_HOME/bin/action_nameCmd.ps1

**Command-line Syntax**
You invoke an action script by specifying the action-name segment of a script's file name in a TCMAN command line.

```
tcman.sh action-name [general_options] [ action-name_options]
```

**Note:** You do not include the Cmd.extension portion of the file name on the TCMAN command line.

TCMAN first looks for action scripts in the instance ($CATALINA_BASE/bin) and then in the core server ($CATALINA_HOME/bin). When it finds the action script, TCMAN passes all command line options and parameters during execution.

**Example**
The following command line runs a script file named viewCmd.sh and passes the -v option and the logs/catalina.out argument.

```
tcman.sh view -v logs/catalina.out
```
Notes

- When TCMAN does not recognize \textit{action-name} as one of the built-in actions, it looks for an action script. If the specified action is not a built-in action or external action script it will exit with an error.

- When the external action script exits, TCMAN uses its return status to either issue an error or to exit with success. Therefore, extension action scripts should always return a specific value according to the platform type.

- Before executing an external action script, TCMAN exports these environment variables for consumption by the action script file:
  - JAVA\_HOME: the root directory of the JAVA JDK (optional)
  - JRE\_HOME: the root directory of the JAVA JRE (optional)
  - CATALINA\_HOME: the root directory of where PAS was installed
  - CATALINA\_BASE: the root directory of the PAS instance being managed (can be the same as CATALINA\_HOME)
  - CATALINA\_TMPDIR: the temporary file directory for temp files
  - CATALINA\_PID: the file where the PAS process-id is stored
  - tcman\_oss shell: The name of the OS and command line shell TCMAN is running in
  - tcman\_instancebase: the name of the new instance base directory when executing a create action
  - tcman\_httpport: the HTTP network port command line option (-p)
  - tcman\_httpsport: the HTTPS network port command line option (-P)
  - tcman\_cfghttpport: the current instance's configured HTTP network port
  - tcman\_cfghttpsport: the current instance's configured HTTPS network port
  - tcman\_cfgshutport: the current instance's shutdown network port (can be -1 if there is no network shutdown port)
  - tcman\_alias: the instance’s alias name
  - tcman\_type: the type of instance (home, service (on Windows), instance)
  - tcman\_secmodel: the security model to apply (default, production, or development)
  - tcman\_umask: the UNIX umask to use when creating new directories and files
  - tcman\_verbose: the state of the TCMAN \texttt{-v} command line option
  - tcman\_debug: The state of the TCMAN \texttt{-g} command line option

Manager actions

This section details the actions available for deploying, running, and monitoring web applications on a server instance.

See also

The \texttt{tcman} command on page 116
List deployed applications (list)

**Purpose**
Display all the web applications that are deployed on an instance.

**Note:** This command may be used whether the instance is online or offline. However, the output differs. When used offline, TCMAN simply shows a list of deployed application directories in the instance's web applications directory. When used online, it provides additional run-time details about the deployed web applications.

To use this action, the Tomcat manager (manager.war) must be deployed on the instance if the instance is online. You can deploy manager.war from $CATALINA_HOME/extras.

**Syntax**

tcman.sh list [general_options] [-u user_id:password]

**Parameters**

**general_options**
Specify one or more of the options that can be used with any TCMAN action. Run tcman.sh help list to see which general options are appropriate.

**-u user_id:password**
Specify a valid user name and password for HTTP Basic access authentication. (The default is -u tomcat:tomcat.)

**Note:** This option is required if the server is online. It is not required if the server is offline.

**Example**
Show the Web applications deployed to acme1 when the instance is online:

```
/psc/acme1/bin/tcman.sh list -u tomcat:tomcat
OK - Listed applications for virtual host localhost
/:running:0:ROOT
:/manager:running:4:manager
:/oemanager:running:0:oemanager
:/oeadapters:running:0:oeabl
```
Show the Web applications deployed to acme1 when the instance is offline:

```
/psc/acme1/bin/tcman.sh list
OK - Listing directories for /psc/acme1/webapps
经理:stopped:0:manager
/oeadapters:stopped:0:oeabl
/oemanager:stopped:0:oemanager
/:stopped:0:ROOT
```

See also
- Display Web application HTTP sessions (sessions) on page 131
- Deploy a Web application (deploy) on page 123
- Undeploy a Web application (undeploy) on page 124
- Start a Web application (enable) on page 128
- Stop a Web application (disable) on page 129
- The tcman command on page 116

Display OS and server information (info)

Purpose
Display server and OS information for a running instance.

To use this action, the Tomcat manager (manager.war) must be deployed on the instance and the instance must be running. You can deploy manager.war from $CATALINA_HOME/extras.

Use the test action to show configuration information about a server that is not running.

Syntax
```
tcman.sh info [general_options] -u user_name:password
```

Parameters

`general_options`

Specify one or more of the general TCMAN options. Run `tcman.sh help info` to see which general options are appropriate.

`-u user_name:password`

Pass a valid user name and a password for HTTP Basic access authentication. (The default is `-u tomcat:tomcat`.)
Example

Display the OS and server information for the running instance named acme1:

```bash
$: /psc/pashome/tcman.sh info -I acme1 -u tomcat:tomcat
OK - Server info
Tomcat Version: Apache Tomcat/7.0.42
OS Name: Linux
OS Version: 2.6.18-164.el5
OS Architecture: amd64
JVM Version: 1.7.0_02-b13
JVM Vendor: Oracle Corporation
```

See also

- Display server, OS, and runtime version information (version) on page 149
- Display detailed server status (status) on page 126
- Test a server configuration (test) on page 150
- The tcman command on page 116
- Display or modify the server features of an instance (feature) on page 137
- The tcman command on page 116

Deploy a Web application (deploy)

Purpose

Deploy a Web application (.war file) to a PAS instance whether the server is running (online) or is not running (offline). TCMAN copies the web application to the server’s web application directory. If the server is online, you must stop and restart it in order to complete the deployment.

Syntax

```bash
tcman.sh deploy [general_options] [-u user_id:password] [-a app_name] war_file_path
```

Parameters

**general_options**

Specify one or more of the general TCMAN options. Run `tcman.sh help deploy` to see which general options are appropriate.

**-u user_id:password**

Specify a valid user name and password for HTTP Basic access authentication.

**Note:** This option is required if the server in online. It is not required if the server is offline.
Specify a name for the web application. If you do not use this option, the application name will be the same as the .war file name.

Specify the location of the web application .war file that you want to deploy.

Example

Deploy and rename oeabl.war (a web application that implements OpenEdge adapters) to the acme1 instance of the core pashome server:

```
/psc/acme1/bin/tcman.sh deploy -a oeadapters /psc/pashome/extras/oeabl.war
OK - deployed /psc/pashome/extras/oeabl.war to local directory /psc/acme1/webapps
```

Note: The $CATALINA_HOME/extras directory (/psc/pashome/extras in the example above) also contains number of instance management applications, including host-manager.war, manager.war, and oemanager.war.

See also

List deployed applications (list) on page 121
Undeploy a Web application (undeploy) on page 124
Reload a Web application (reload) on page 125
Start a Web application (enable) on page 128
Stop a Web application (disable) on page 129
The tcman command on page 116

Undeploy a Web application (undeploy)

Purpose

Remove a Web application from running (online) or stopped (offline) instances. If the instance's autodeploy option is off, you must stop and restart a running server to complete removal. Note that the autodeploy option is set in the .../conf/appserver.properties file and is off by default.

Syntax

```
tcman.sh undeploy [general_options] [-u user_id:password] app_name
```

Parameters

general_options

Specify one or more of the options that can be used with any TCMAN action. Run tcman.sh help undeploy to see which general options are appropriate.
Specify a valid user name and password for HTTP Basic access authentication. (The default is -u tomcat:tomcat.) This option is required if you are accessing an online instance.

**app_name**

Specify the name of the web application to remove.

**Example**

Remove the oemanager application from the acme1 instance:

```
/psc/acme1/bin/tcman.sh undeploy -u tomcat:tomcat oemanager
OK - Undeployed application at context path /oemanager
```

**See also**

- Deploy a Web application (deploy) on page 123
- List deployed applications (list) on page 121
- Reload a Web application (reload) on page 125
- The tcman command on page 116

---

**Reload a Web application (reload)**

**Purpose**

Restart a deployed, running Web application so that the application can pick up changes to its classes or libraries.

To use this action, the Tomcat manager (manager.war) must be deployed on the instance and the instance must be running. You can deploy manager.war from $CATALINA_HOME/extras.

**Note:** The reload action does not reload the web application's web.xml file. To begin using changes to web.xml, you must stop and restart the web application.

**Syntax**

```
tcman.sh reload [general_options] -u user_id:password app_name
```

**Parameters**

- **general_options**

  Specify one or more of the options that can be used with any TCMAN action. Run tcman.sh help reload to see which general options are appropriate.
-u user_id:password

Specify a valid user name and password for HTTP Basic access authentication. (The default is -u tomcat:tomcat.)

Note: This option is required if the server in online. It is not required if the server is offline.

app_name

Specify the name of the web application to restart.

Example

Reload the oemanager web application running on the acme1 instance:

```
    /psc/acme1/bin tcman.sh reload -u tomcat:tomcat oemanager
    OK - Reloaded application at context path /oemanager
```

See also

Start a Web application (enable) on page 128
Stop a Web application (disable) on page 129
List deployed applications (list) on page 121
The tcman command on page 116

Display detailed server status (status)

Purpose

List information from the core server’s memory, including web application statistics. Information includes memory pool usage, connector thread status, and connector status. Output is in XML format. (Note that redirecting the output to an XML viewer makes it more readable.)

To use this action, the Tomcat manager (manager.war) must be deployed on the instance and the instance must be running. You can deploy manager.war from SCATALINA_HOME/extras.

Syntax

```
tcman.sh status [general_options] -u user_name:password [-f]
```

Parameters

general_options

Specify one or more of the general TCMAN options. Run tcman.sh help status to see which general options are appropriate.
Pass a valid user name and a password for HTTP Basic access authentication. (The default is `-u tomcat:tomcat`.)

```
-f
```

Return full status information.

**Example**

Display core server's memory and web application statistics and use `xmllint` to format for readability:

```
$: tcman.sh status -u tomcat:tomcat | xmllint --format -
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/xsl" href="/manager/xform.xsl" ?>
<status>
  <jvm>
    <memory free="453196832" total="520028160" max="1051394048"/>
    <memorypool name="PS Eden Space" type="Heap memory" usageInit="50331648" usageCommitted="469762048" usageMax="1006632960" usageUsed="63861584"/>
    <memorypool name="PS Old Gen" type="Heap memory" usageInit="469762048" usageCommitted="469762048" usageMax="1006632960" usageUsed="63861584"/>
    <memorypool name="PS Survivor Space" type="Heap memory" usageInit="8388608" usageCommitted="1507328" usageMax="1507328" usageUsed="1444184"/>
    <memorypool name="Code Cache" type="Non-heap memory" usageInit="2555904" usageCommitted="2555904" usageMax="2555904" usageUsed="2555904"/>
    <memorypool name="PS Perm Gen" type="Non-heap memory" usageInit="67108864" usageCommitted="67108864" usageMax="67108864" usageUsed="47406400"/>
  </jvm>
  <connector name="http-bio-8601">
    <threadInfo maxThreads="150" currentThreadCount="0" currentThreadsBusy="0"/>
    <requestInfo maxTime="0" processingTime="0" requestCount="0" errorCount="0" bytesReceived="0" bytesSent="0"/>
    <workers/>
  </connector>
  <connector name="http-bio-8501">
    <threadInfo maxThreads="300" currentThreadCount="10" currentThreadsBusy="1"/>
    <requestInfo maxTime="2008" processingTime="2116" requestCount="10" errorCount="0" bytesReceived="0" bytesSent="5838"/>
    <workers>
      <worker stage="S" requestProcessingTime="2" requestBytesSent="0" requestBytesReceived="0" remoteAddr="127.0.0.1" virtualHost="localhost" method="GET" currentUri="/manager/status" currentQueryString="XML=true" protocol="HTTP/1.1"/>
    </workers>
  </connector>
</status>
```

**See also**

Display OS and server information (info) on page 122
Display memory leaks (leaks) on page 127
The tcman command on page 116

**Display memory leaks (leaks)**

**Purpose**

List Web applications with potential memory leaks.
To use this action, the Tomcat manager (manager.war) must be deployed on the instance and the instance must be running. You can deploy manager.war from $CATALINA_HOME/extras.

**Syntax**

```
tcman.sh leaks [general_options] -u user_name:password
```

**Parameters**

*general_options*

Specify one or more of the general TCMAN options. Run `tcman.sh help leaks` to see which general options are appropriate.

*-u user_name:password*

Pass a valid user name and a password for HTTP Basic access authentication. (The default is `-u tomcat:tomcat`.)

**Example**

Display memory leaks for web applications deployed on the acme1 server instance:

```
/psc/acme1/bin/tcman.sh leaks -u tomcat:tomcat
OK - Found potential memory leaks in the following applications:
/warehouse
```

**See also**

- List deployed applications (list) on page 121
- The tcman command on page 116

**Start a Web application (enable)**

**Purpose**

Start a web application that is deployed but not running.

To use this action, the Tomcat manager (manager.war) must be deployed on the instance and the instance must be running. You can deploy manager.war from $CATALINA_HOME/extras.

**Syntax**

```
tcman.sh enable [general_options] -u user_id:password app_name
```
Parameters

general_options

Specify one or more of the options that can be used with any TCMAN action. Run `tcman.sh help start` to see which general options are appropriate.

`-u user_id:password`

Specify a valid user name and password for HTTP Basic access authentication. (The default is `-u tomcat:tomcat`.)

`app_name`

Specify the name of the web application to start.

Note: To start the ROOT web application, you can specify `/` or `ROOT`.

Example

Start the `oeabl` application deployed on the `acme1` instance:

```
tcman.sh enable -u tomcat:tomcat oeabl
OK - Started application at context path /oeabl
```

See also

- Stop a Web application (disable) on page 129
- The `tcman` command on page 116
- The `tcman` command on page 116

Stop a Web application (disable)

Purpose

Stop a running Web application.

To use this action, the Tomcat manager (`manager.war`) must be deployed on the instance and the instance must be running. You can deploy `manager.war` from `$CATALINA_HOME/extras`.

Syntax

```
tcman.sh disable [general_options] [-u user_id:password] app_name
```
Parameters

general_options

Specify one or more of the general TCMAN options. Run `tcman.sh help disable` to see which general options are appropriate.

- `u user_id:password`

  Specify a valid user name and password for HTTP Basic access authentication. (The default is `-u tomcat:tomcat`.)

app_name

Specify the name of the web application to disable.

Note: To disable the ROOT web application, you can specify `/` or `ROOT`.

Example title

Disable the `oeabl` application running on the `acme1` instance:

```
/psc/acme1/bin/tcman.sh disable -u tomcat:tomcat oeabl
OK - Stopped application at context path /oeabl
```

See also

Start a Web application (enable) on page 128
The tcman command on page 116

Display global server resources (resources)

Purpose

List the global resources used by the core server.

To use this action, the Tomcat manager (`manager.war`) must be deployed on the instance and the instance must be running. You can deploy `manager.war` from `$CATALINA_HOME/extras`.

Syntax

```
tcman.sh resources [general_options] -u user_name:password
```

Parameters

general_options

Specify one or more of the general TCMAN options. Run `tcman.sh help resources` to see which general options are appropriate.
-u user_name:password

Pass a valid user name and a password for HTTP Basic access authentication.
(The default is -u tomcat:tomcat.)

Example

Display global resources for the running instance, acme1:

```
$: /psc/acme1/bin/tcman.sh resources -u tomcat:tomcat
OK - Listed global resources of all types
ServiceRegistry/ServiceRegistryFactory:com.progress.appserv.services.naming.ServiceRegistry
UserDatabase:org.apache.catalina.users.MemoryUserDatabase
```

Display Web application HTTP sessions (sessions)

Purpose

Display how many sessions are active for the specified Web application, categorized by their duration.
To use this action, the Tomcat manager (manager.war) must be deployed on the instance and the instance must be running. You can deploy manager.war from $CATALINA_HOME/extras.

Syntax

```
tcman.sh sessions [general_options] -u user_id:password app_name
```

Parameters

general_options

Specify one or more of the options that can be used with any TCMAN action.

- -u user_id:password

Specify a valid user name and password for HTTP Basic access authentication. (The default is -u tomcat:tomcat.)

app_name

Specify the name of the web application to analyze for session information.
Example

Show the active sessions for the manager application deployed on the acme1 instance:

```
/psc/acme1/bin/tcman.sh sessions -u tomcat:tomcat manager
OK - Session information for application at context path /manager
Default maximum session inactive interval 30 minutes
<1 minutes: 1 sessions
8 - <9 minutes: 2 sessions
9 - <10 minutes: 1 sessions
```

See also

List deployed applications (list) on page 121

Server actions

This section details the actions available for creating and monitoring server instances.

See also

The tcman command on page 116

Create an instance (create)

Purpose

Create a new instance of the core PAS server by running this action from /bin directory of the core server ( $CATALINA_HOME/bin/tcman.sh create).

Syntax

```
tcman.sh create [general_options] [-f] [-p port_num] [-P port_num]
[-s port_num] [-j port_num] [-m uid:pwd] [-W pathname] [-N instance_name]
[-U user_id] [-G group_id] [-Z {prod | dev}] base_path
```

Parameters

general_options

Specify one or more of the general TCMAN options. Run tcman.sh help create to see which general options are appropriate.

-f

Copy all deployed web application archives (.war files) from $CATALINA_HOME to the new instance.
-p port_num

Specify the TCP port that listens for HTTP messages. The default is 8080.

-P port_num

Specify the TCP port that listens for HTTPS messages. The default is 8443.

-s port_num

Specify the TCP port to use to stop an instance. On Windows systems, you must specify a shutdown port. On UNIX, shutdown ports are optional.

-j port_num

Specify the TCP port that listens for AJP13 messages (an Apache protocol for handling requests from a web server to an application server). The default is 8009.

-m uid:pwd

Specify a user name and password that will be required to access Tomcat container-level security, which includes the manager and oemanager web applications. Replaces the defaults (tomcat:tomcat) in /conf/tomcat-users.xml.

-W pathname

Specify the directory where web applications will be deployed. The default is $CATALINA_BASE/webapps.

-N instance_alias

Specify an alias for the instance. If you do not specify an alias, the instance name will be the name of the directory where the instance is created.

Note:
All instances are automatically registered for tracking when they are created. However, for tracking to function, the instance name must not contain spaces or any of the following characters: "[ . # | ] $ ? + = { / , ]"

-U user_id

Specify the user-id of the owner of all the files and directories of the instance. The default is the user-id of the current process. -G must be specified if you use this option.

-G group_id

Specify the group-id of the owner of all the files and directories of the instance. The default is the group-id of the current process. -U must be specified if you use this option.

-Z {dev | prod}

Specify the security model of the instance to development (dev) or secure (prod).
A typical use of this option is for testing web applications in a secure server environment before packaging and deploying.
**Note:** The -Z prod option does not create a production server. To actually create a production server, you must have a production server license.

**base_path**

Specify the pathname where you will create the instance.

**Example**

Create an instance of /psc/pashome in /psc/acme1:

```
$: /psc/pashome/bin/tcman.sh create -p 8501 -P 8601 -s 8701 /psc/acme1
Server instance acme1 created at /psc/acme1
```

**See also**

- Delete an instance (delete) on page 134
- Display and manage an instance's configuration (config) on page 135
- Register an instance for tracking (register) on page 142
- Start an instance (start) on page 147
- Stop an instance (stop) on page 148

**Delete an instance (delete)**

**Purpose**

Remove the directory tree and all of the files in an instance. Alias tracking is disabled for servers that are removed.

To execute this action, the instance cannot be running.

**Note:** You cannot recover any files or directories removed by the delete action. Backup anything you want to save before launching this action.

Also note that you cannot use delete to remove the installed, root server (`$CATALINA_HOME`).

**Syntax**

```
tcman.sh delete [general_options] [-y] [base_path | alias_name]
```

**Parameters**

**general_options**

Specify one or more of the general TCMAN options. Run `tcman.sh help delete` to see which general options are appropriate.
Delete everything without prompting for confirmation.

**base_path**

Specify the pathname of the instance that you intend to delete.

**alias_name**

Refer to the instance that you intend to delete by its alias rather than its pathname.

**Example**

Delete the instance of **pashome** that was created in **/psc/acme3**:

```
$: /psc/pashome/bin/tcman.sh delete /psc/acme3
The following directory tree will be removed permanently:
  ( WARNING all deployed web applications will be DELETED!! )
/PAS/wrkdir/acme3
/PAS/wrkdir/acme3/conf
/PAS/wrkdir/acme3/temp
/PAS/wrkdir/acme3/common
/PAS/wrkdir/acme3/common/lib
/PAS/wrkdir/acme3/logs
/PAS/wrkdir/acme3/webapps
/PAS/wrkdir/acme3/webapps/ROOT
/PAS/wrkdir/acme3/webapps/ROOT/WEB-INF
/PAS/wrkdir/acme3/webapps/ROOT/WEB-INF/adapters
/PAS/wrkdir/acme3/webapps/ROOT/WEB-INF/adapters/soap
/PAS/wrkdir/acme3/webapps/ROOT/WEB-INF/classes
/PAS/wrkdir/acme3/webapps/ROOT/WEB-INF/classes/com
/PAS/wrkdir/acme3/webapps/ROOT/WEB-INF/classes/com/progress
/PAS/wrkdir/acme3/webapps/ROOT/WEB-INF/classes/com/progress/appserv
/PAS/wrkdir/acme3/work
/PAS/wrkdir/acme3/bin
Type 'yes' to continue
yes
Delete operation complete
server removed at /PAS/wrkdir/acme3
```

**See also**

- Create an instance (create) on page 132
- Display server instances (instances) on page 140
- The tcman command on page 116

**Display and manage an instance's configuration (config)**

**Purpose**

View, add, update, or delete the property values specified in **../conf/appserver.properties** and in **../conf/catalina.properties**.
When you run `tcman.sh config` with no parameters, it displays the core Tomcat server's configuration, and all the properties in both `.../conf/appserver.properties` and `.../conf/jvm.properties`. Note, however, that you can only view `jvm.properties`. You cannot modify its contents with the `config` action.

**Syntax**

```
tcman.sh config [general_options]
[ prop_name | prop_name=value | +prop_name=value | ~prop_name ]
```

**Parameters**

`general_options`

Specify one or more of the general TCMAN options. Run `tcman.sh help config` to see which general options are appropriate.

`prop_name`

Display the specified property and its value.

`prop_name=value`

Set the value of a property that exists in `.../conf/appserver.properties`.

`+prop_name=value`

Add a new property to `.../conf/appserver.properties` and set its value.

`~prop_name`

Remove the specified property from `.../conf/appserver.properties`.

**Examples**

Show the configuration and properties of `acme1`, an instance of the core server, `pashome`:

```
$: /psc/acme1/bin/tcman.sh config
Using CATALINA_BASE: /psc/acme1
Using CATALINA_HOME: /psc/pashome
Using CATALINA_TMPDIR: /psc/acme1/temp
Using JRE_HOME: /tools/linuxx86_64/java64/jdk1.7.0_02/
Using CLASSPATH: /psc/pashome/bin/bootstrap.jar:/psc/pashome/bin/tomcat-juli.jar
Using CATALINA_PID: /psc/acme1/temp/catalina.pid
Server version: Apache Tomcat/7.0.42
Server built: Jul 2 2013 08:57:41
Server number: 7.0.42.0
OS Name: Linux
OS Version: 2.6.18-164.el5
Architecture: amd64
JVM Version: 1.7.0_02-b13
...
```
Display the value of a single property:

$: /psc/acme1/bin/tcman.sh config psc.as.http.port
psc.as.http.port=8501

Update the value of a property that exists in the appserver.properties file and then check the value:

$: /psc/acme1/bin/tcman.sh config psc.as.http.port=6543
$: tcman.sh config psc.as.http.port
psc.as.http.port=6543

Add a new property/value pair to the appserver.properties file and check the value:

$: /psc/acme1/bin/tcman.sh config +my.home.dir=/home/jarhead
$: tcman.sh config my.home.dir
my.home.dir=/home/jarhead

Update the server certificate in the catalina.properties file (see https://docs.oracle.com/cd/E19879-01/821-0185/ablqz/index.html for information about generating, exporting, and downloading a new server certificate):

$: /psc/acme1/bin/tcman.sh config psc.as.https.keyalias=myNewCert

Remove a property/value pair from the appserver.properties file and check if deletion was successful:

$: /psc/acme1/bin/tcman.sh config ~my.home.dir
$: tcman.sh config my.home.dir
Property does not exist - my.home.dir

Caution: There are no restrictions to property removal. The server will be unable to start if you remove a property required by conf/server.xml.

Notes

- All property names are case sensitive.
- You cannot enter multiple property names (prop_name) on the command line to view, update, or add properties to the appserver.properties file.
- You cannot use the config action to update existing values or add new values to the jvm.properties file.

Display or modify the server features of an instance (feature)

Purpose

View, enable, or disable the server features contained in the /conf/server.xml file of an instance.
When you run `tcman.sh feature` with no parameters, it displays a list of the features (and their current status) that you can enable or disable. You can also display the status of a single server feature. After viewing the status of a feature, you can use `tcman.sh feature` to change its setting.

**Syntax**

```
tcman.sh feature [general_options] [feature_name=[on | off]]
```

**Parameters**

*general_options*

Specify one or more of the general TCMAN options. Run `tcman.sh help feature` to see which general options are appropriate.

*feature_name*

Specify one of the features defined in an instance's `conf/server.xml` file. Running `tcman.sh feature` without *feature_name* displays a list of all the features.

*on*

Enables the named feature.

*off*

Disables the named feature.

**Example**

Display the list of server feature settings for `acme1`, enable AJP13 (Apache JServ Protocol. version 1.3), and verify that the feature is enabled:

```
$: /psc/acme1/bin/tcman.sh feature
SecurityListener=off
JMXLifecycle=off
PSCRegistry=on
HTTP=onHTTPS=on
AJP13=off
Cluster=off
UserDatabase=on
JAASRealm=off
LDAPRealm=off
PASInstrument=off
RemoteHostValve=on
RemoteAddrValve=onSingleSignOn=on
AccessLog=on
CrawlerSessionManager=on
StuckSessionValve=on

$: /psc/acme1/bin/tcman.sh feature AJP13=on

$: /psc/acme1/bin/tcman.sh feature AJP13
AJP13=on
```
Notes

• Server features for instances are set in $CATALINA_BASE/conf/server.xml. You can change feature status by manually editing this file. However, it is safer to use `tcman.sh feature` to avoid corrupting the file with erroneous entries.

• Run `tcman.sh feature` when the instance is offline.

See also
Display and manage an instance’s configuration (config) on page 135
Stop an instance (stop) on page 148
The tcman command on page 116

Clean up or archive server log files (clean)

Purpose
Truncate, move, or delete the log files located in the /logs directory of the core server or instance. If the server is running, clean truncates log files to zero length. If the server is not running, clean deletes the log files from the file system.

You have the option to save log files to a subdirectory of /logs.

Syntax

```
tcman.sh clean [general_options] [-A]
```

Parameters

general_options

Specify one or more of the general TCMAN options. Run `tcman.sh help clean` to see which general options are appropriate.

-A

Archive log files to a subdirectory of $CATALINA_BASE/logs. The directory is automatically named with a month-day-year-second (MM-DD-YYYY-ss) time-stamp format. If the server is not running, the files in $CATALINA_BASE/logs are deleted.

Example

Archive the log files of acme1, an instance of the core server pashome, and save to a file:

```
/sc/pashome/tcman.sh clean -I acme1 -A
```

See also

The tcman command on page 116
Display server instances (instances)

Purpose
Show the names and locations of the instances created from the PAS installed in $CATALINA_HOME by displaying the contents of the file where instances are registered for tracking.

By default, instances are registered for tracking $CATALINA_HOME/conf/instances.{windows|.unix}. The file name extension indicates the OS platform where the PAS server is installed.

Syntax

tcman.sh instances [general_options]

Parameters

general_options

Specify one or more of the general TCMAN options. Run tcman.sh help instances to see which general options are appropriate.

Output format

The following is the format of the output from a TCMAN instances action:

alias-name | full-file-path | type | state

alias-name

The user-defined name for the instance.

full-file-path

The location, in the OS file system, of the instance's root directory.

type

The designation of the server instance type (for example: instance, service, ...).

state

An indication of the instance's validity. OK is returned for a valid server and invalid is returned for a corrupted or non-existant server.
Example

Display the instances of the core server installed in /psc/pashome:

```
/psc/pashome/bin/tcman.sh instances
acme1 | /psc/wrk/acme1 | instance | ok
acme2 | /psc/wrk/acme2 | instance | ok
```

Notes

- By default, instances are registered when you execute a `$CATALINA_HOME/bin/tcman{.sh|.bat} create` action, which automatically adds instance entries to an `instances` file. TCMAN removes instance entries from the file when you execute a delete action.
  
  You can manually add or remove instance entries from `instances` by using the `register` or `unregister` actions.

- By default, the name and location of the file where instances are registered is `$CATALINA_HOME/conf/instances.{windows|.unix}`.

  You can change the location of the instance registration file by adding and setting the `psc.as.instdir` property in the `appserver.properties` file. Use the TCMAN config action as in the following example:

  ```
tcman.sh config '+psc.as.instdir=PATH'
  ```

  where `PATH` is a path name or an environment variable.

  You can also change the location and/or name of instance registration files by setting the environment variables, `PAS_AS_INSTANCE_DIR`, and `PAS_AS_INSTANCE_FILE`.

See also

Register an instance for tracking (register) on page 142
Stop tracking an instance (unregister) on page 151
The tcman command on page 116

List process ids (plist)

Purpose

List the process ids for all the processes that are running under an instance.

Syntax

```
tcman.sh plist [general_options] [-f]
```
Parameters

**general_options**

Specify one or more of the options that can be used with any TCMAN action. Run `tcman.sh help plist` to see which general options are appropriate.

**-f**

Display verbose output. The output is indented and uses the plus (+) character to indicate parent-child relationships.

**Examples**

Display process id's for the running instance, `acme1` using the `-v` and `-f` options:

```text
/psc/acme1/bin/tcman.sh plist -v
info: showing process ids for server 5942
  5942 5963 5975 5988 6001 6015

/psc/acme1/bin/tcman.sh plist -f
5942
+5963
+5975
+5988
+6001
+6015
```

**Notes**

The plist action is useful for administrative tasks such as:

- Checking to see if processes persist after an instance is stopped.
- Checking if an multi-session agent process has started and is available
- Checking if an instance is running. Output is 0 if it is not running.
- Using the output (which is easily parseable) in administrative scripts.

**See also**

Show Windows process information (showproc) on page 146

**Register an instance for tracking (register)**

**Purpose**

Register an instance for tracking purposes.
Note:
Instances are automatically registered for tracking when you execute a create action. You use the register action to restart tracking on instances after tracking was stopped.

A typical use for unregistering and then re-registering an instance is to make configuration changes when moving instances from one location (core server) to another. The register action enables tracking and also updates the value of CATALINA_HOME in all of the executable scripts in the instance’s /bin directory to refer to the new core server.

Syntax

```
tcman.sh register alias_name instance_path
```

Parameters

`alias_name`

Specify a meaningful name for the instance. The alias name must be unique in the instances file.

`instance_path`

Specify the OS file system path to where the instance exists. This value will be expanded into a fully qualified OS directory path and will be verified to exist.

Example

Track `test1`, which is an alias for the instance `/psc/acmel`:

```
/psc/pashome/bin/tcman.sh register test1 /psc/acmel
```

Notes

When you register an instance for tracking or create a new instance with the `create` command, an entry is created in the core Progress Application Server’s `$CATALINA_HOME/conf/instances.[unix|windows]` file.

The `instances.[unix|windows]` file is a simple text file, which can be manually edited (with care) in the event that it becomes out of date. The format for entries is:

```
instance_name = base_path
```

An `instances.unix` file uses Unix OS file path syntax (forward slashes), and an `instances.windows` file uses Windows OS file path syntax (backslashes) to specify `base_path`.

Also note that in an instances file:

- Any line starting with a pound-sign (`#`) is a comment line.
- Blank lines are skipped.
Instance names cannot contain spaces or any of the following characters: ":[ . # | ] $ ? + = / , ]"

See also
Stop tracking an instance (unregister) on page 151
The tcman command on page 116

Register and manage an instance as a Windows service (service)

Purpose
(Windows only) Registers or unregisters an instance as a Windows service. After an instance is registered, you can start, stop, or check the status of the service with this action.

Note: Before you register an instance as a Windows service, you must install JDK 1.8 (for example, jdk1.8.0_66) and set the environment variable JAVA_HOME=C:\Program Files\Java\jdk1.8.0_66.

Syntax

```
tcman.bat service [general_options] alias_name { register | unregister | start
| stop | status }
```

Parameters

general_options

Specify one or more of the general TCMAN options. Run `tcman.bat help service` to see which general options are appropriate.

alias_name

A required parameter that specifies the name of a PAS instance that was created using `tcman create`.

register

Create a new Windows service that runs the named PAS instance `alias_name`.

Set the PR_DISPLAYNAME and/or PR_DESCRIPTION variables to change the display name and description of the PAS instance service that appears in the Windows Service utility (Services tab of the Task Manager). The defaults for these variables are:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>PR_DISPLAYNAME</td>
<td>Progress Application Server <code>alias_name</code></td>
</tr>
<tr>
<td>PR_DESCRIPTION</td>
<td>Progress Application Server (Tomcat 7) – <a href="http://www.progress.com">http://www.progress.com</a></td>
</tr>
</tbody>
</table>
Set these variables before you register the instance. For example, if you wanted to change the defaults for `oepas1`:

```
set PR_DISPLAYNAME=PAS ROOT Server
set PR_DESCRIPTION=Progress Application Server
tcman service oepas1 register
```

 unregister

Delete the Windows service that runs the named PAS instance `alias_name`.

start

Start the Windows service that corresponds to the named PAS instance `alias_name`. The Windows service may also be started using the Windows service console or the SC command line utility.

stop

Stop the Windows service that corresponds to the named PAS instance `alias_name`. The Windows service may also be stopped using the Windows service console or the SC command line utility.

status

The registration status of the Windows service corresponding to the named PAS instance `alias_name`. The Windows service's status may be monitored using the Windows service console or SC command line utility.

Example

Register the default instance `oepas1` as a Windows, then start, check status, stop, and unregister:

```
tcman service oepas1 register
  oepas1 service is registered
  tcman service oepas1 start
  oepas1 started
  tcman service oepas1 status
    Service oepas1 is running
  tcman service oepas1 stop
    oepas1 is stopped
  tcman service oepas1 unregister
    oepas1 is unregistered
```

Note

Be sure that the instance is not running before you attempt to register/unregister it.
Show Windows process information (showproc)

**Purpose**

(Windows only) Show information about the process specified by a process id.

**Syntax**

```
tcman.bat showproc [general_options] [process-id]
```

**Parameters**

*general_options*

Specify one or more of the options that can be used with any TCMAN action. Run `tcman.sh help showproc` to see which general options are appropriate.

*process-id*

The numerical identifier of a Windows process. You can obtain a list of process ids by running the TCMAN `plist` action.

**Examples**

Display process id's for the running instance, `acmel`, then specify process ids to show detailed information.

```
/psc/acmel/bin/tcman.bat plist -v
info: showing process ids for server with window title 13332
13332 14240

/psc/acmel/bin/tcman.bat showproc 13332
ProcesName : java
SessionId : 2
StartTime : 10/04/2015 16:29:42
Threads : 26
TotalProcessorTime : 00:00:19.9213277
UserProcessTime :
CPU (seconds) : 19.9213277
Description : Java(TM) Platform SE binary
Path : C:\Progress\OpenEdge\jdk\bin\java.exe

/psc/acmel/bin/tcman.bat showproc 14240
ProcesName : _mproapsv
SessionId : 2
StartTime : 10/04/2015 16:29:54
Threads : 7
TotalProcessorTime : 00:00:00.3744024
UserProcessTime :
CPU (seconds) : 0.3744024
Description : OpenEdge AppServer (Multi-thread)
Path : C:\Progress\OpenEdge\bin\_mproapsv.exe
```
See also
List process ids (plist) on page 141

Start an instance (start)

Purpose
Start an instance of a PAS, optionally in debug mode.

Syntax

```
tcman.sh start [general_options] [-D | -J]
```

Parameters

`general_options`
Specify one or more of the general TCMAN options. Run `tcman.sh help start` to see which general options are appropriate.

`-D`
Start the server in Tomcat debug mode. `-D` overrides the `-J` option.

`-J`
Start the server in debug mode using the JDPA (Java Platform Debugger Architecture) APIs for debugging. `-J` cannot be used if the `-D` option is specified.

Before you run a server with the `-J` option, you must define a port for the JDPA debugger by setting the `JDPA_ADDRESS` environment variable to a unique TCP network port number.

Example
Start the server in `/psc/acme1`, which is an instance of the core server in `/psc/pashome`:

```
psc/acme1/bin/tcman.sh start
Using CATALINA_BASE: /psc/acme1
Using CATALINA_HOME: /psc/pashome
Using CATALINA_TMPDIR: /psc/acme1/temp
Using JRE_HOME: /tools/linuxx86_64/java64/jdk1.7.0_02/
Using CLASSPATH: /psc/pashome/bin/bootstrap.jar:/psc/pashome/bin/tomcat-juli.jar
Using CATALINA_PID: /psc/acme1/temp/catalina.pid
```

Notes

- When the TCMAN utility starts the server, it verifies the creation of the OS process and then records the server's process-id in a `.pid` file. The location of the `.pid` file is:
You can obtain the process id of a server by running the TCMAN `env` action.

**See also**
- Stop an instance (stop) on page 148
- The `tcman` command on page 116

## Stop an instance (stop)

### Purpose
Stop a running instance, either gracefully or forcibly.

### Note:
TCMAN supports stopping a server instance that is not configured with a shutdown port.

On UNIX platforms stopping the running server instance is accomplished by sending a UNIX signal to the PAS process. Therefore, the administrator's process must have the UNIX permissions to signal the PAS process. On Windows platforms, the instance is identified using an OS process id that is used to stop server processes.

### Syntax

```
tcman.sh stop [general_options] [ -F [-w seconds]]
```

### Parameters

#### general_options

Specify one or more of the general TCMAN options. Run `tcman.sh help stop` to see which general options are appropriate.

- `-F`

  Kill the server process if it does not stop after a default wait time (5 seconds on UNIX, 10 seconds on Windows). Change the default wait interval by using the `-w` option.

- `-w ` seconds`

  Optionally specify the number of seconds to wait before killing a server process.
Example

Stop the server in `/psc/acme1`, which is an instance of the core server in `/psc/pashome`:

```
/psc/acme1/bin/tcman.sh stop
Using CATALINA_BASE: /psc/acme1
Using CATALINA_HOME: /psc/pashome
Using CATALINA_TMPDIR: /psc/acme1/temp
Using JRE_HOME: /tools/linuxx86_64/java64/jdk1.7.0_02/
Using CLASSPATH: /psc/pashome/bin/bootstrap.jar:/psc/pashome/bin/tomcat-juli.jar
Using CATALINA_PID: /psc/acme1/temp/catalina.pid
```

Notes

• TCMAN supports stopping a server instance that is not configured with a shutdown port.

On UNIX platforms stopping the running server instance is accomplished by sending a UNIX signal to the PAS process. Therefore, the administrator's process must have the UNIX permissions to signal the PAS process. On Windows platforms, the instance is identified using an OS process id that is used to stop server processes.

The following is an example a message you would see after a forced shut down with no shut down port:

```
SEVERE: No shutdown port configured. Shut down server through OS signal.
Server not shut down.
Killing Tomcat with the PID: 14230
```

• Process ids are stored in the following locations:

<table>
<thead>
<tr>
<th>OS</th>
<th>PID File Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIX</td>
<td>$CATALINA_BASE/temp/catalina-instance_name.pid</td>
</tr>
<tr>
<td>Windows</td>
<td>$CATALINA_BASE\logs\catalina-instance_name.pid</td>
</tr>
</tbody>
</table>

• You can also obtain the process id of a server by running the TCMAN `env` action.

See also

- Start an instance (start) on page 147
- The tcman command on page 116

Display server, OS, and runtime version information (version)

Purpose

Show the Apache Tomcat runtime version and OS information for an instance.

To execute this action, the instance cannot be running
Chapter 10: TCMAN Reference

Syntax

tcman.sh version [general_options]

Parameters

general_options

Specify one or more of the general TCMAN options. Run tcman.sh help version to see which general options are appropriate.

Example

Display the server and runtime information for acme1, an instance of the core server installed in /psc/pashome:

$: /psc/pashome/bin/tcman.sh version -I acme1
Using CATALINA_BASE: /psc/acme1
Using CATALINA_HOME: /psc/pashome
Using CATALINA_TMPDIR: /psc/acme1/temp
Using JRE_HOME: /tools/linuxx86_64/java64/jdk1.7.0_02/
Using CLASSPATH:
/psc/pashome/bin/bootstrap.jar:/users/doc/agarbacz/psc/pashome/bin/tomcat-juli.jar
Using CATALINA_PID: /psc/acme1/temp/catalina.pid
Server version: Apache Tomcat/7.0.42
Server built: Jul 2 2013 08:57:41
Server number: 7.0.42.0
OS Name: Linux
OS Version: 2.6.18-164.el5
Architecture: amd64
JVM Version: 1.7.0_02-b13
JVM Vendor: Oracle Corporation

See also

Display OS and server information (info) on page 122
The tcman command on page 116

Test a server configuration (test)

Purpose

Displays information on the configuration and environment of an instance. It also displays information about error conditions.

The test action starts a server (instance), loads all the configuration files, and then displays information. The instance is stopped, exiting gracefully even if there is an error condition.

To execute this action, the instance cannot be running.
Syntax

tcman.sh test [general_options]

Parameters

general_options

Specify one or more of the general TCMAN options. Run tcman.sh help test to see which general options are appropriate.

Example

Run a test of the configuration of acmel, which is an instance of the core server installed at /psc/pashome:

```bash
$: /psc/pashome/bin/tcman.sh -I acmel test
Using CATALINA_BASE: /psc/acmel
Using CATALINA_HOME: /psc/pashome
Using CATALINA_TMPDIR: /psc/acmel/temp
Using JRE_HOME: /tools/linuxx86_64/java64/jdk1.7.0_02/
Using CLASSPATH: /psc/pashome/bin/bootstrap.jar:/psc/pashome/bin/tomcat-juli.jar
Using CATALINA_PID: /psc/acmel/temp/catalina.pid
...```

Notes

The test action is particularly useful for testing to verify that a server will start and run properly after you make changes to configuration and properties files.

See also

The tcman command on page 116

Stop tracking an instance (unregister)

Purpose

Stop tracking an instance by removing the instance's entry from the $CATALINA_HOME/conf/instances.[unix|windows] file.

Note:

You use the register action to restart tracking on instances after tracking was stopped with unregister.

A typical use for unregistering and then re-registering an instance, is to make configuration changes when moving instances from one location, or core server, to another. The register action not only enables tracking, it also updates the value of CATALINA_HOME in all of the executable scripts in the instance’s /bin directory to refer to the new core server.
Syntax

tcman.sh unregister alias_name

Parameters

alias_name

Specify the alias name of the instance that you want to stop tracking. The alias name must exist in an instances.[unix|windows] file.

Example

Stop tracking test1, which is an instance of /psc/pashome:

`/psc/pashome/bin/tcman.sh unregister test1`

See also

- Register an instance for tracking (register) on page 142
- The tcman command on page 116

Create a Tomcat worker configuration file (workers)

Purpose

Create a preliminary worker.properties file that supports the configuration of supporting servers (workers) for a Progress Application Server (PAS) instance.

In the Apache Reference Guide, a worker is defined as an "instance that is waiting to execute servlets or any other content on behalf of some web server." In the context of the Progress Application Server, a worker is a server that is called by a PAS instance to perform a specific task. Typically, you would define worker instances to manage proxies, load balancing, clusters, or status monitoring. (For links to information on this functionality, see the Apache Tomcat Documentation Index.) There are probably other situations where you could improve the performance of a server instance by configuring worker instances to handle specific processing tasks.

In Apache Tomcat, workers are configured in a worker.properties file. The protocol implemented for communication between servers and workers is the Apache JServ Protocol (version 1.3, referred to as AJP13).

In TCMAN, the workers action adds the definitions of registered PAS instances to the content of the $CATALINA_HOME/extras/workers.template file and puts the result in $CATALINA_HOME/temp/worker.properties. The template file supplies a set of common directives that are referenced by all of the defined PAS instances. Individual instance definitions contain only the properties that are unique to the instance, such as the AJP13 network connection port. (See Table 17: worker.properties example on page 154.)

The /temp/worker.properties created by the workers action is a preliminary configuration file that you will probably need to modify to implement your deployment. See The Apache Tomcat Connector-Reference Guide for more information about configuring workers.
Syntax

```
$CATALINA_HOME/tcman.sh workers [general_options] [worker_list]
```

Parameters

general_options

Specify one or more of the general TCMAN options. Run `tcman.sh help worker` to see which general options are appropriate.

worker_list

A comma separated list of instance names and/or keywords. The keywords are:

<table>
<thead>
<tr>
<th>keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>Include an instance that has been implemented as a status server</td>
</tr>
<tr>
<td>lb</td>
<td>Include an instance that has been implemented as a load balance server</td>
</tr>
<tr>
<td>home</td>
<td>Include the CATALINA_HOME core server</td>
</tr>
<tr>
<td>all</td>
<td>Include all registered instances</td>
</tr>
</tbody>
</table>

If no `worker_list` is specified, the default worker list (all instances registered to CATALINA_HOME) will be added. If no instances have been created, then the default `worker_list` is CATALINA_HOME.

Examples

Assume there are:

- Two registered instances (`piw1` and `piw2`) that serve Web applications
- A Tomcat load balancer instance (`jklb`) that distributes the workload between `piw1` and `piw2`
- A status instance (`jkstatus`) that is used to monitor the runtime status of `piw1` and `piw2`

The following are examples of worker-lists showing various combinations of keywords and instances, and the resulting content in `$CATALINA_HOME/temp/worker.properties`:

<table>
<thead>
<tr>
<th>worker-list</th>
<th>Resulting content in worker.properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>blank</td>
<td>Default entries from <code>worker.template</code> plus entries for <code>piw1</code> and <code>piw2</code></td>
</tr>
<tr>
<td>piw1</td>
<td>Default entries from <code>worker.template</code> plus an entry for <code>piw1</code></td>
</tr>
<tr>
<td>all</td>
<td>Default entries from <code>worker.template</code> plus entries for <code>piw1</code> and <code>piw2</code></td>
</tr>
<tr>
<td>home</td>
<td>Default entries from <code>worker.template</code> plus an entry for core server (CATALINA_HOME)</td>
</tr>
</tbody>
</table>
### Table 17: worker.properties example

<table>
<thead>
<tr>
<th>worker-list</th>
<th>Resulting content in worker.properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>home, all</td>
<td>Default entries from <code>worker.template</code> plus an entry for core server, (CATALINA_HOME), and entries for piw1 and piw2</td>
</tr>
<tr>
<td>lb, status</td>
<td>Default entries from <code>worker.template</code> plus entries for jklb, jstatus, piw1, and piw2</td>
</tr>
<tr>
<td>lb, status, home, all</td>
<td>Default entries from <code>worker.template</code> plus entries for jklb, jstatus, the core server (CATALINA_HOME), piw1, and piw2</td>
</tr>
</tbody>
</table>

**Notes**

- The `tcmn workers` action must be run from the PAS installation's `$CATALINA_HOME/bin` directory.
- The `/extras/workers.template` file can be modified to adjust existing properties or to add additional static information. However, you cannot replace the common properties with a unique set of properties for each defined server.

# General actions

This section details the actions available for displaying help and server runtime environment information.

**See also**

*The tcman command* on page 116

## Display help (help)

### Purpose

Display summary or detailed help for all TCMAN actions, property names, and server features.

### Syntax

```bash
tcman.sh help [action|property|feature]
```

### Parameters

**action**

Show the syntax and options of the specified action. If no action is specified, show a list of all actions and the general options.

**property**

Show the settings for specified property.

**feature**

Show if the specified feature is enabled or disabled.
Example

Display the usage help for the `create` action:

```bash
$: tcman.sh help create
    instance-opts:
        [-s <shutdown-port>]
        [-P <https-port>]
        [-j <ajp13-port>]
        [-W <web-apps-dir>]
        [-N <inst-alias-name>]
        [-U <file-owner> -G <file-group>]

general options:
    -u uid:pwd  pass uid and pwd for HTTP BASIC authentication
    -v           print verbose output
    -M url      override the CATALINA_BASE manager's URL with
                <http|https://<host>:<port>/<mgr-app>
    -B           override CATALINA_BASE environment setting
    -n           debug run action but do not execute changes
```

Display runtime environment information (env)

Purpose

Show details about a server’s state.

Syntax

```bash
tcman.sh env [general_options] [keyword]
```

Parameters

general_options

Specify one or more of the general TCMAN options. Run `tcman.sh help env` to see which general options are appropriate.

keyword

Specify one or more keywords that represent the name of the state that you want to view. If no keyword is specified, then all of the state information is displayed.

Keywords include:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>running</td>
<td>Indicate if a server is running (1) or not running (0).</td>
</tr>
<tr>
<td>mgrurl</td>
<td>Display the URL of the manager application.</td>
</tr>
<tr>
<td>Keyword</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>type</td>
<td>Display the server type.</td>
</tr>
<tr>
<td>alias</td>
<td>Display the server’s alias.</td>
</tr>
<tr>
<td>parent</td>
<td>Display the pathname of the parent of an instance.</td>
</tr>
<tr>
<td>tracking</td>
<td>Indicate if tracking is on (1) or off (0).</td>
</tr>
<tr>
<td>http</td>
<td>Display the server’s http port number.</td>
</tr>
<tr>
<td>https</td>
<td>Display the server’s https port number.</td>
</tr>
<tr>
<td>shut</td>
<td>Display the server’s shutdown port number. A value of -1 indicates that there is no shutdown port.</td>
</tr>
<tr>
<td>pid</td>
<td>Display the server’s process id. A hyphen (−) indicates that the server is not running.</td>
</tr>
</tbody>
</table>

### Example

Display all of the state information for the instance created in `/psc/acme1`:

```
/psc/acme1/bin/tscman.sh env
catalina home: /psc/pashome
catalina base: /psc/acme1
java home: /tools/linuxx86_64/java64/jdk1.7.0_02/
jre home:
manager http port: 8501
manager https port: 8601
manager shut port: 8701
manager URL: http://localhost:8501/manager
config type: instance
config alias: acme1
config parent: /psc/pashome
server running: 0
instance tracking: 1
instance file: /psc/pashome/conf/instances.unix
server process-id: -
```
The oemanager.war Java Web application provides a REST API for remote management and monitoring of ABL applications deployed on a Progress Application Server for OpenEdge instance. It is similar to the administration API supported by the Tomcat Web server’s JMX interface, but it uses JSON input/output payloads.

**Requirements**

To enable REST API access, either oemanager.war or tcmanager.war must be deployed.

**Examples**

The examples in this section show the use of curl, which is a common open software utility used in command lines or scripts to transfer data. You can, however, employ the PAS for OpenEdge REST management APIs with many other tools and utilities. For more information about curl, see: http://curl.haxx.se/docs/.

For details, see the following topics:

- Session management
- Agent management
- Transport management
- Application management

**Session management**

This section describes the REST services that are available for managing Progress Application Server for OpenEdge sessions.
Get session information

Description
Query an ABL application to get a list of all its sessions including information on client and agent connections.

HTTP Operation
GET

URI
//host_name:port/oemanager/applications/ABL_app_name/sessions

Media type
application/vnd.progress+json

Response codes
200 Success
404 Resource Not Found
500 Unexpected Server Error

Command-line example
curl -X GET -v http://localhost:16680/oemanager/applications/oepas1/sessions

Request body example
NA

Response body example
{
  "operation":"GET CLIENT SESSIONS",
  "result":
  {
    "OEABLSession":
      [
        
      ]
  
}
Get client information

Description
Query an ABL application to get information on client connections.

HTTP Operation
GET

URI
//host_name:port/oemanager/applications/ABL_app_name/clients[?timeoutms=n]
timeoutms

The period of time, specified in milliseconds (ranging from 1 to $n$), to report on client connections, including hung clients.

Media type

application/vnd.progress+json

Response codes

200 Success
500 Unexpected Server Error

Command-line examples

curl -X GET -v http://localhost:16680/oemanager/applications/oepas1/clients


Request body example

NA

Response body example

```json
{
"operation":"GET CLIENT CONNECTIONS",
"result":
{
"ClientConnection":
[
{
"elapsedTimeMs":427133,
"requestProcedure":"ngasPing.p",
"requestUrl":
"http://\localhost.com:16680/apsv",
"clientName":"172.21.75.148",
"httpSessionId":"732277EE12485DDEC3A937DD30D7EF3267B3FE669C76.pas1",
"executerThreadId":"catalina-exec-8",
"reqStartTimeStr":"2014-09-17T08:41:48.138-0400",
"requestID":"ngasPing.p",
"adapterType":"HTTP",
"sessionId":"Iphz681UQKexuHibyy6S+A"
}
]
,"outcome":"SUCCESS",
"errmsg":"
,"versionStr":"PASOE 11.5.0",
"versionNo":1
}
```

Get current request information

Description

Query an ABL application to get a list of all the requests currently in process.
HTTP Operation

GET

URI

//host_name:port/oemanager/applications/ABL_app_name/requests

Media type

application/vnd.progress+json

Response codes

200 Success
500 Unexpected Server Error

Command-line example

curl -X GET -v http://localhost:16680/oemanager/applications/oepas1/requests

Request body example

NA

Response body example

```json
{
"operation":"GET ACTIVE REQUESTS ON SERVER",
"result":
{
"Request":
[
{
"requestElapsedTime":1889,
"requestStartTimeStamp":"2014-09-17T10:11:26.934-0400",
"requestState":"RUNNING",
"requestID":"xfJqPACGU4jqE4pd0Auzwg",
"sessionID":"8DvBtOp1RHijAj7NcSAQRw"
}
],
"outcome":"SUCCESS",
"errmsg":"",
"versionStr":"PASOE 11.5.0",
"versionNo":1
}
```

Get configuration properties

Description

Query the conf/openedge.properties file to get the configuration properties of an application. The information is specified in the AppServer.SessMgr.App_name section of openedge.properties.
HTTP Operation
GET

URI
//host_name:port/oemanager/applications/App_name/properties

Media type
application/vnd.progress+json

Response codes
200 Success
404 Resource Not Found
500 Unexpected Server Error

Command-line example
curl -X GET -v http://localhost:16680/oemanager/applications/oepas1/properties

Request body example
NA

Response body example
{
    "operation":"GET SESSION-MGR PROPERTIES",
    "result":
    {
        "agentLogThreshold":"0",
        "requestWaitTimeout":"15000",
        "agentLogEntryTypes":"ASPlumbing, DB.Connects",
        "idleAgentTimeout":"300000",
        "idleConnectionTimeout":"300000",
        "socketTimeout":"3000",
        "agentListenerTimeout":"300000",
        "applications":"oepas1",
        "numInitialAgents":"1",
        "agentLogFile":"/wrk/pas1/logs/oepas1.agent.log",
        "collectMetrics":"1",
        "agentExecFile":"/localhost/dlc/bin/_mproapsv",
        "idleResourceTimeout":"0",
        "allowRuntimeUpdates":"0",
        "connectionWaitTimeout":"3000",
        "maxAgents":"10",
        "publishDir":"/wrk",
        "maxConnectionsPerAgent":"16",
        "idleSessionTimeout":"300000",
        "agentStartupParam":"-T /wrk/pas1/temp",
        "statusEnabled":"1",
        "maxABLSessionsPerAgent":"200",
        "agentNumLogFiles":"3",
        "agentLoggingLevel":"3",
        "ipver":"IPv4"
    },
    "outcome":"SUCCESS",
    "errmsg":"
    "versionStr":"PASOE 11.5.0",
}
Update configuration properties

Description

HTTP Operation
PUT

URI
//host_name:port/oemanager/applications/App_name/properties

Media type
application/vnd.progress+json

Response codes
200 Success
500 Unexpected Server Error

Command-line example

curl -X PUT -v http://localhost:16680/oemanager/applications/oepas1/properties
-d '{"agentLogThreshold":"1"}' -H "Content-Type: application/vnd.progress+json"

Request body example
{
    "operation":"SET BROKER PROPERTIES",
    "result":
    {
        "agentLogThreshold":"1",
        "requestWaitTimeout":"15000",
        "agentLogEntryTypes":"ASPlumbing,DB.Connects",
        "idleAgentTimeout":"300000",
        "idleConnectionTimeout":"300000",
        "socketTimeout":"3000",
        "agentListenerTimeout":"300000",
        "applications":null,
        "numInitialAgents":null,
        "agentLogFile":null,
        "collectMetrics":null,
        "agentExecFile":null,
        "idleResourceTimeout":null,
        "allowRuntimeUpdates":null,
        "connectionWaitTimeout":null,
        "maxAgents":null,
        "publishDir":null
    }
}
"maxConnectionsPerAgent":"16",
"idleSessionTimeout":"300000",
"agentStartupParam":"-T \scratch\prmundra\17Sept\wrk\pas1\temp",
"statusEnabled":"1",
"maxABLSessionsPerAgent":"200",
"agentNumLogFile":"3",
"agentLoggingLevel":"3",
"ipver":"IPv4"
},
"outcome":"SUCCESS",
"errmsg":",
"versionStr":"PASOE 11.5.0",
"versionNo":1
}

Response body examples

On success:

{
  "operation":"SET SESSION-MGR PROPERTIES",
  "result":"NA",
  "outcome":"SUCCESS",
  "errmsg":",
  "versionStr":"PASOE 11.5.0",
  "versionNo":1
}

On failure:

{
  "Operation":"SET SESSION-MGR PROPERTIES",
  "Outcome":"FAILURE",
  "ErrorMsg":"Invalid property name: "InvalidPropName". (PRPMsg009)"
}

Get runtime metrics

Description
Query the session manager for runtime metrics of an ABL application.

HTTP Operation
GET

URI
//host_name:port/oemanager/applications/ABL_app_name/metrics

Media type
application/vnd.progress+json

Response codes
200 Success
500 Unexpected Server Error
Command-line example

```
curl -X GET -v http://localhost:16680/oemanager/applications/oepas1/metrics
```

Request body example

NA

Response body example

```
{
  "operation": "GET SESSION-MGR METRICS",
  "result": {
    "writeErrors": 0,
    "concurrentConnectedClients": 0,
    "maxQueueDepth": 0,
    "maxConcurrentClients": 0,
    "readErrors": 0,
    "reserveConnectionTimeouts": 0,
    "timesQueued": 0,
    "Writes": 0,
    "Requests": 0,
    "Reads": 0,
    "avgQueueDepth": 0,
    "type": "OE BROKER",
    "accessTime": "2014-09-17T10:26:12.693-04:00"
  },
  "outcome": "SUCCESS",
  "errmsg": "",
  "versionStr": "PASOE 11.5.0",
  "versionNo": 1
}
```

Reset runtime metrics

**Description**
Reset runtime metrics of the session manager.

**HTTP Operation**
DELETE

**URI**
```
//host_name:port/oemanager/applications/ABL_app_name/metrics
```

**Media type**
application/vnd.progress+json

**Response codes**
- 200 Success
- 500 Unexpected Server Error
Cancel a request

Description
Cancel the request identified by a Request ID and a Session ID. You can obtain Request IDs and Session IDs by executing the `oemanager/applications/ABL_app_name/sessions` service.

HTTP Operation
DELETE

URI
`//host_name:port/oemanager/applications/App_name/requests?requestID=value&sessionID=value`

Media type
NA

Response codes
200 Success
500 Unexpected Server Error

Command-line example
```
curl -X DELETE -v "http://localhost:16680/oemanager/applications/pas1
/requests?requestID=z9WHoRgGY4fnExrFWKWlw&sessionID=G1TTNRPT3KiSOhkbyeSow"
```

Request body example
NA

Response body example
```
{
    "operation":"CANCEL ACTIVE REQUEST ON SERVER",
}
```
Terminate a session

Description
Terminate the session identified by the specified sessionID.

HTTP Operation
DELETE

URI
//host_name:port/oemanager/applications/App_name/sessions?sessionID=value&terminateOpt={ 0 | 1 }

where a terminateOpt value of 0 causes a graceful termination and a value of 1 causes a forced termination.

Media type
NA

Response codes
200 Success
500 Unexpected Server Error

Command-line example

Request body example
NA

Response body example
{
   "operation":"TERMINATE SESSIONS",
   "result":"",
   "outcome":"FAILURE",
   "errmsg":"Session",
   "versionStr":"PASOE 11.5.0",
   "versionNo":1
}"
Agent management

This section describes the REST services that are available for managing Progress Application Server for OpenEdge agents.

See also
List agents on page 170
Get agent status information on page 171
Get session metrics on page 172
Get connection information on page 173
Get request information on page 174
Get configuration properties on page 178
Update configuration properties on page 179
Stop an agent on page 180

List agents

Description
Get a list of all the PAS Agents available to an ABL application.

HTTP Operation
GET

URI
//host_name:port/oemanager/applications/ABL_app_name/agents

Media type
application/vnd.progress+json

Response codes
200 Success
500 Unexpected Server Error

Command-line example

curl -X GET -v http://localhost:16680/oemanager/applications/pas1/agents

Request body example
NA
Response body example

```json
{
    "result": {
        "agents": [
            {
                "agentId": "AG-sCIVXeFSQYmcb7RcHo10Zw",
                "pid": "12924",
                "state": "AVAILABLE"
            }
        ],
        "operation": "GET AGENTS",
        "versionStr": "PASOE 11.5.0",
        "versionNo": 1,
        "outcome": "SUCCESS",
        "errmsg": ""
    }
}
```

Get agent status information

**Description**
Get a summary of sessions, connections and requests for an agent of an ABL application.

**HTTP Operation**
GET

**URI**
```
//host_name:port/oemanager/applications/ABL_app_name/agents/agentID/status
```

**Media type**
application/vnd.progress+json

**Response codes**
- 200 Success
- 500 Unexpected Server Error

**Command-line example**
```
curl -X GET -v http://localhost:16680/oemanager/applications/pas1/agents/AG-sCIVXeFSQYmcb7RcHo10Zw/status
```

**Request body example**
NA

**Response body example**
```
{
    "result": {
        "threads": 6,
        "sessions": 5,
```
Get session metrics

Description
Get session metrics for an agent of an ABL application.

HTTP Operation
GET

URI
//host_name:port/oemanager/applications/ABL_app_name/agents/agentID/sessions

Media type
application/vnd.progress+json

Response codes
200 Success
500 Unexpected Server Error

Command-line example
curl -X GET -v http://localhost:16680/oemanager/applications/pas1/agents/AG-sCIVXeFSQYmcb7RcHo10Zw/sessions

Request body example
NA

Response body example
{
  "result": {
    "AgentSession": [
      {
        "SessionId": 1,
        "SessionState": "LISTENER",
        "StartTime": "2014-09-22T11:07:49.744",
        "EndTime": null,
        "ThreadId": 2,
        "ConnectionId": null,
        "SessionExternalState": 0
      },
    ]
  }
}
Get connection information

Description
Get information on the connections of an agent of an ABL application.

HTTP Operation
GET

URI
//host_name:port/oemanager/applications/ABL_app_name/agents/agentID/connections

Media type
application/vnd.progress+json
Response codes

200 Success
500 Unexpected Server Error

Command-line example

curl -X GET -v
http://localhost:16680/oemanager/applications/pas1/agents/AG-sCIVeFSQmcb7RcHo10Zw/connections

Request body example

NA

Response body example

{
    "result": {
        "AgentConnection": [ 
            { 
                "ConnectionId": 7, 
                "ConnectionState": "LISTENER", 
                "SessionId": -1 
            }, 
            { 
                "ConnectionId": 10, 
                "ConnectionState": "LISTENER", 
                "SessionId": -1 
            }, 
            { 
                "ConnectionId": 11, 
                "ConnectionState": "ADMINISTRATOR", 
                "SessionId": -1 
            }, 
            { 
                "ConnectionId": 19, 
                "ConnectionState": "ADMINISTRATOR", 
                "SessionId": -1 
            }, 
            { 
                "ConnectionId": 22, 
                "ConnectionState": "REQUEST", 
                "SessionId": -1 
            } 
        ], 
        "operation": "GET CONNECTIONS METRICS", 
        "versionStr": "PASOE 11.5.0", 
        "versionNo": 1, 
        "outcome": "SUCCESS", 
        "errmsg": ""
    }
}

Get request information

Description
Get request information for an agent of an ABL application.

HTTP Operation
GET
Agent management

**URI**

//host_name:port/oemanager/applications/ABL_app_name/agents/agentID/requests

**Media type**

application/vnd.progress+json

**Response codes**

200 Success
500 Unexpected Server Error

**Command-line example**

curl -X GET -v http://localhost:16680/oemanager/applications/pas1/agents/AG-sCIVXeFSQYmb7RcHo10Zw/requests

**Request body example**

NA

**Response body example**

```
{
   "result": {
      "AgentRequest": [
         {
            "RequestProcName": "OpenEdge.Server.Admin.AgentInfo:getAgentThreadData",
            "SessionId": 3,
            "ConnectionId": 19,
            "StartTime": "2014-09-22T11:07:49.744",
            "EndTime": "2014-09-22T11:07:49.745",
            "RequestNum": 0
         },
         {
            "RequestProcName": "OpenEdge.Server.Admin.AgentInfo:getAgentSessionData",
            "SessionId": 3,
            "ConnectionId": 19,
            "StartTime": "2014-09-22T11:07:49.756",
            "EndTime": "2014-09-22T11:07:49.757",
            "RequestNum": 1
         },
         {
            "RequestProcName": "OpenEdge.Server.Admin.AgentInfo:getAgentConnectionData",
            "SessionId": 3,
            "ConnectionId": 19,
            "StartTime": "2014-09-22T11:07:49.759",
            "EndTime": "2014-09-22T11:07:49.760",
            "RequestNum": 2
         },
         {
            "RequestProcName": "OpenEdge.Server.Admin.AgentInfo:getAgentRequestData",
            "SessionId": 3,
            "ConnectionId": 19,
            "StartTime": "2014-09-22T11:07:49.762",
            "EndTime": "2014-09-22T11:07:49.763",
            "RequestNum": 3
         },
         {
            "RequestProcName": "OpenEdge.Server.Admin.AgentInfo:getAgentThreadData",
            "SessionId": 3,
            "ConnectionId": 19,
```
"StartTime": "2014-09-22T11:07:55.812",
"EndTime": "2014-09-22T11:07:55.813",
"RequestNum": 4
},
{
"RequestProcName": "OpenEdge.Server.Admin.AgentInfo:getAgentSessionData",
"SessionId": 3,
"ConnectionId": 19,
"StartTime": "2014-09-22T11:07:55.814",
"EndTime": "2014-09-22T11:07:55.815",
"RequestNum": 5
},
{
"RequestProcName": "OpenEdge.Server.Admin.AgentInfo:getAgentConnectionData",
"SessionId": 3,
"ConnectionId": 19,
"StartTime": "2014-09-22T11:07:55.817",
"EndTime": "2014-09-22T11:07:55.817",
"RequestNum": 6
},
{
"RequestProcName": "OpenEdge.Server.Admin.AgentInfo:getAgentRequestData",
"SessionId": 3,
"ConnectionId": 19,
"StartTime": "2014-09-22T11:07:55.819",
"EndTime": "2014-09-22T11:07:55.819",
"RequestNum": 7
},
{
"RequestProcName": "OpenEdge.Server.Admin.AgentInfo:getAgentThreadData",
"SessionId": 3,
"ConnectionId": 19,
"EndTime": "2014-09-22T11:09:22.653",
"RequestNum": 8
},
{
"RequestProcName": "OpenEdge.Server.Admin.AgentInfo:getAgentSessionData",
"SessionId": 3,
"ConnectionId": 19,
"EndTime": "2014-09-22T11:09:22.655",
"RequestNum": 9
},
{
"RequestProcName": "OpenEdge.Server.Admin.AgentInfo:getAgentConnectionData",
"SessionId": 3,
"ConnectionId": 19,
"EndTime": "2014-09-22T11:09:22.657",
"RequestNum": 10
},
{
"RequestProcName": "OpenEdge.Server.Admin.AgentInfo:getAgentRequestData",
"SessionId": 3,
"ConnectionId": 19,
"RequestNum": 11
},
{
"RequestProcName": "OpenEdge.Server.Admin.AgentInfo:getAgentThreadData",
"SessionId": 3,
"ConnectionId": 19,
"StartTime": "2014-09-22T11:09:23.001",
"RequestNum": 12
},
"RequestProcName": "OpenEdge.Server.Admin.AgentInfo:getAgentConnectionData",
"SessionId": 3,
"ConnectionId": 19,
"StartTime": "2014-09-22T11:09:23.007",
"EndTime": "2014-09-22T11:09:23.007",
"RequestNum": 14
},

"RequestProcName": "OpenEdge.Server.Admin.AgentInfo:getAgentRequestData",
"SessionId": 3,
"ConnectionId": 19,
"StartTime": "2014-09-22T11:09:23.024",
"EndTime": "2014-09-22T11:09:23.025",
"RequestNum": 15
},

"RequestProcName": "OpenEdge.Server.Admin.AgentInfo:getAgentSessionData",
"SessionId": 3,
"ConnectionId": 19,
"StartTime": "2014-09-22T11:10:23.689",
"EndTime": "2014-09-22T11:10:23.690",
"RequestNum": 16
},

"RequestProcName": "OpenEdge.Server.Admin.AgentInfo:getAgentSessionData",
"SessionId": 3,
"ConnectionId": 19,
"StartTime": "2014-09-22T11:10:34.078",
"EndTime": "2014-09-22T11:10:34.079",
"RequestNum": 17
},

"RequestProcName": "OpenEdge.Server.Admin.AgentInfo:getAgentSessionData",
"SessionId": 3,
"ConnectionId": 19,
"StartTime": "2014-09-22T11:10:34.389",
"EndTime": "2014-09-22T11:10:34.390",
"RequestNum": 18
},

"RequestProcName": "OpenEdge.Server.Admin.AgentInfo:getAgentConnectionData",
"SessionId": 3,
"ConnectionId": 19,
"StartTime": "2014-09-22T11:12:03.324",
"EndTime": "2014-09-22T11:12:03.324",
"RequestNum": 19
},

"RequestProcName": "OpenEdge.Server.Admin.AgentInfo:getAgentConnectionData",
"SessionId": 3,
"ConnectionId": 19,
"StartTime": "2014-09-22T11:12:44.784",
"EndTime": "2014-09-22T11:12:44.785",
"RequestNum": 20
},

"RequestProcName": "OpenEdge.Server.Admin.AgentInfo:getAgentConnectionData",
"SessionId": 3,
"ConnectionId": 19,
"StartTime": "2014-09-22T11:12:45.131",
"EndTime": "2014-09-22T11:12:45.132",
Get configuration properties

Description
Get configuration information (properties) of agents for an ABL application.

HTTP Operation
GET

URI
//host_name:port/oemanager/applications/App_name/agents/properties

Media type
application/vnd.progress+json

Response codes
200 Success
500 Unexpected Server Error
Command-line example

curl -X GET -v http://localhost:16680/oemanager/applications/pas1/agents/properties

Request body example
NA

Response body example

```json
{
  "result": {
    "workDir": "/wrk/pas1/work",
    "keyAliasPasswd": "",
    "sslAlgorithms": "",
    "agentMinPort": "62002",
    "infoVersion": "9610",
    "sessionDisconnProc": "",
    "sslEnable": "0",
    "applications": "pas1",
    "PROPATH": "/wrk/pas1/openedge,/dlc/tty",
    "collectStatsData": "0",
    "agentStartupProcParam": "",
    "agentShutdownProc": "",
    "collectMetrics": "1",
    "sessionStartupProcParam": "",
    "collectStatsData": "0",
    "sessionDeactivateProc": "",
    "keyStorePath": ".\keys\",
    "sessionActivateProc": "",
    "allowRuntimeUpdates": "0",
    "sessionConnectProc": "",
    "sessionTimeout": "180",
    "sessionStartupProc": "",
    "statusEnabled": "1",
    "agentMaxPort": "62202",
    "keyAlias": "",
    "uuid": "http://localhost:9999/oepas1",
    "agentStartupProc": ""
  },
  "operation": "GET AGENT PROPERTIES",
  "versionStr": "PASOE 11.5.0",
  "versionNo": 1,
  "outcome": "SUCCESS",
  "errmsg": ""
}
```

Update configuration properties

Description
Update the configuration properties in `conf/openedge.properties` for agents for an ABL application.

HTTP Operation
PUT
**URI**

//host_name:port/oemanager/applications/App_name/agents/properties

\{ "prop_name"=value[, "prop_name"=value[, ...]] \} 

"prop_name"=value[, prop_name"=value[, ...]]

Set one or more properties expressed as name/value pairs in a comma-separated list. See the /conf/openedge.properties.README file for a description of properties and their possible values.

**Media type**

application/vnd.progress+json

**Response codes**

200 Success

500 Unexpected Server Error

**Command-line example**

curl -X PUT -v http://localhost:16680/oemanager/applications/pas1/agents/properties 
-d '{"collectMetrics":0}' -H "Content-Type: application/vnd.progress+json"

**Request body example**

{""collectMetrics":0}

**Response body example**

NA

**Stop an agent**

**Description**

Stop an agent belonging to an ABL application. Note that you identify the agent by the `agentID`, which can be obtained by executing /oemanager/applications/ABL_app_name/agents.

**HTTP Operation**

DELETE

**URI**

//host_name:port/oemanager/applications/App_name/agents/agentID

**Media type**

application/vnd.progress+json
Response codes

200 Success
500 Unexpected Server Error

Command-line example

curl -X DELETE -v
http://localhost:16680/oemanager/applications/oepas1/agents/AG-sCIVXeFSQYmcb7RcHo10Zw
-H "Content-Type: application/vnd.progress+json"

Request body example

NA

Response body example

{
   "result": {
      "agentID": "AG-sCIVXeFSQYmcb7RcHo10Zw"
   },
   "operation": "STOP AGENT",
   "versionStr": "PASOE 11.5.0",
   "versionNo": 1,
   "outcome": "SUCCESS",
   "errmsg": ""
}

Transport management

This section describes the REST services that are available for managing PAS for OpenEdge SOAP, REST, and APSV transports.

See also

Get transport configuration properties (SOAP/REST) on page 182
Update properties of a transport (SOAP/REST) on page 183
Enable or disable a transport (SOAP/REST) on page 185
Enable or disable a transport (APSV) on page 186
Get runtime metrics of a transport (SOAP/REST) on page 187
Get runtime metrics of a transport (APSV) on page 188
Reset runtime metrics for a transport (SOAP/REST) on page 189
Reset runtime metrics of a transport (APSV) on page 190
List deployed services (SOAP/REST) on page 191
Get information about a service (SOAP/REST) on page 193
Get configuration properties of a service (SOAP) on page 194
Deploy a new service (REST) on page 195
Deploy a new service (SOAP) on page 196
Update runtime properties of a service (SOAP) on page 197
Get runtime metrics for a service (SOAP) on page 198
Reset runtime metrics for a service (SOAP) on page 200
Enable or disable a service (SOAP) on page 200
Get transport configuration properties (SOAP/REST)

Description
Get configuration properties of a REST or SOAP transport of a Web application belonging to an ABL application.

HTTP Operation
GET

URI

//host_name:port/oemanager/applications/
ABL_app_name/webapps/Web_app_name/transports/{soap|rest}/properties

Media type
application/vnd.progress+json

Request body
NA

Response codes
200 Success
500 Unexpected Server Error

Command-line example (SOAP)
curl -X GET -v
http://localhost:16680/oemanager/server/oeservices/ROOT/transports/soap/properties

Response body example (SOAP)
{"result":
{"idleSessionTimeout":0,
"waitIfBusy":0,
"connectionLifetime":0,
"requestWaitTimeout":-1,
"staleO4GLOBjectTimeout":0,
"clientASKActivityTimeout":60,
"initialSessions":1,
"clientASKResponseTimeout":60,
"minIdleConnections":0,
"maxSessions":0,
"minSessions":1,
"appServerKeepalive":"denyClientASK,allowServerASK",
"serviceServerKeepalive":0,
"serviceFaultLevel":2
},
"outcome":"SUCCESS",
"versionNo":1,
"errormsg":"NA",
"versionStr":
"PASOE 11.5.0",
"operation":"GET SOAP TRANSPORT PROPERTIES"
}

Command-line example (REST)

curl -X GET -v
http://localhost:16680/oemanager/server/oeservices/ROOT/transports/rest/properties

Response body example (REST)

{
 "result": {
   "appServerKeepalive": "denyClientASK,allowServerASK",
   "idleSessionTimeout": 0,
   "minSessions": 1,
   "requestWaitTimeout": -1,
   "initialSessions": 1,
   "waitIfBusy": 0,
   "staleO4GLObjectTimeout": 0,
   "minIdleConnections": 0,
   "connectionLifetime": 0,
   "serviceFaultLevel": 2,
   "clientASKResponseTimeout": 60,
   "maxSessions": 0,
   "clientASKActivityTimeout": 60
 },
 "operation": "GET REST TRANSPORT PROPERTIES",
 "versionStr": "PASOE 11.5.0",
 "versionNo": 1,
 "outcome": "SUCCESS",
 "errmsg": "

Update properties of a transport (SOAP/REST)

Description
Update the properties of a SOAP or REST transport of a Web application belonging to an ABL application.

HTTP Operation
PUT

URI
//host_name:port/oemanager/applications/
ABL_app_name/webapps/Web_app_name/transports/{soap|rest}/properties

Media type
application/vnd.progress+json

Response codes
200 Success
500 Unexpected Server Error
Command-line example (SOAP)

curl -X PUT -v
http://localhost:16680/oemanager/applications/pas1/webapps/ROOT/transports/soap/properties
-H "Content-Type: application/vnd.progress+json" -d '{"waitIfBusy":1}'

Request body example (SOAP)

{"waitIfBusy":1}

Response body example (SOAP)

{"result":
{"idleSessionTimeout":0,
 "waitIfBusy":1,
 "connectionLifetime":0,
 "requestWaitTimeout":-1,
 "staleO4GLObjectTimeout":0,
 "clientASKActivityTimeout":60,
 "initialSessions":1,
 "clientASKResponseTimeout":60,
 "minIdleConnections":0,
 "maxSessions":0,
 "minSessions":1,
 "appServerKeepalive":"denyClientASK,allowServerASK",
 "serviceFaultLevel":2
 },
 "outcome":"SUCCESS",
 "versionNo":1,
 "errmsg":"NA",
 "versionStr":
 "PASOE 11.5.0",
 "operation":"SET SOAP TRANSPORT PROPERTIES"
}

Command-line example (REST)

curl -X PUT -v
http://localhost:16680/oemanager/server/oeservices/ROOT/transports/rest/properties
-H "Content-Type: application/vnd.progress+json" -d '{"maxSessions":1}'

Request body example (REST)

{"maxSessions":1}

Response body example (REST)

{"operation":"SET REST TRANSPORT PROPERTIES",
 "result":{"appServerKeepalive":"denyClientASK,allowServerASK",
 "idleSessionTimeout":0,
 "minSessions":1,
 "requestWaitTimeout":-1,
 "initialSessions":1,
 "waitIfBusy":0,
 "staleO4GLObjectTimeout":0,
 "minIdleConnections":0,
 "connectionLifetime":0,
 "serviceFaultLevel":2,
 "clientASKResponseTimeout":60,
 "maxSessions":1,
 "clientASKActivityTimeout":60"}
Enable or disable a transport (SOAP/REST)

Description
Enable or disable the REST or SOAP transport of a Web application belonging to an ABL application.

HTTP Operation
PUT

URI
//host_name:port/oemanager/applications/
ABL_app_name/webapps/Web_app_name/transports/\{soap\|rest\}/state

Media type
application/vnd.progress+json

Request body
{
    "STATE" : "\{ENABLED \| DISABLED\}"
}

Response codes
200 Success
500 Unexpected Server Error

Command-line example (SOAP)
curl -X PUT -v 
http://localhost:16680/oemanager/applications/pas1/webapps/ROOT/transports/soap/state
-d '("state":"ENABLED")' -H "Content-Type: application/vnd.progress+json"

Response body example (SOAP)
{"state":"ENABLED"}

Command-line example (REST)
curl -X PUT -v 
http://localhost:16680/oemanager/applications/pas1/webapps/ROOT/transports/rest/state
-d '("state":"DISABLED")' -H "Content-Type: application/vnd.progress+json"
Response body example (REST)
{"state":"DISABLED"}

Enable or disable a transport (APSV)

Description
Enable or disable the APSV transport of a Web application belonging to an ABL application.

HTTP Operation
PUT

URI
//host_name:port/oemanager/applications/ABL_app_name/webapps/Web_app_name/transports/state

Media type
application/vnd.progress+json

Response codes
200 Success
500 Unexpected Server Error

Command-line example

curl -X DELETE -d '{"state":"DISABLED"}' -v
http://localhost:16680/oemanager/applications/pas1/webapps/ROOT/transports/state

Request body example
{"state":"DISABLED"}

Response body example
{
  "operation":"SET APSV TRANSPORT STATE",
  "result":"NA",
  "versionNo":1,
  "versionStr":"PASOE 11.5.0",
  "outcome":"SUCCESS",
  "errmsg":"NA"
}
Get runtime metrics of a transport (SOAP/REST)

Description
Get runtime metrics for a REST or SOAP transport of a Web application deployed in an ABL application.

HTTP Operation
GET

URI

//host_name:port/oemanager/applications/
ABL_app_name/webapps/Web_app_name/transports/{soap|rest}/metrics

Media type
application/vnd.progress+json

Request body
NA

Response codes
200 Success
500 Unexpected Server Error

Command-line example (SOAP)
curl -x GET -v

Response body example (SOAP)
{"result":
{"serviceDisabled":0,
"soapProcessorErrors":0,
"httpRequestErrors":0,
"httpRequests":0,
"statusRequest":0,
"methodNotAllowedErrors":0,
"soapRequests":0,
"successfulSoapRequests":0,
"wsdlRequest":0,
"activeRequests":0,
"urlNotFoundErrors":0,
"type":"OE_SOAP_TRANSPORT",
"startTime":"2014-07-17T08:40:03.422-04:00",
"accessTime":"2014-07-17T10:34:00.295-04:00"},
"outcome":"SUCCESS",
"versionNo":1,
"errmsg":"NA",
"versionStr":"PASOE 11.5.0"}
"operation":"GET OE_SOAP_TRANSPORT METRICS"
}

Command-line example (REST)

Response body example (REST)
{
  "result":{
    "successfulConnectRequests":0,
    "serviceNotFound":0,
    "numRequests":0,
    "serviceUnavailableRequests":0,
    "statusRequests":0,
    "runRequest":0,
    "connectRequests":0,
    "successfulRequests":0,
    "ExpressionError":0,
    "failedRequests":0,
    "successfulRunRequest":0,
    "type":"OE_REST_TRANSPORT",
    "accessTime":"2014-07-17T08:40:58.315-04:00"
  },
  "outcome":"SUCCESS",
  "versionNo":1,
  "errmsg":"NA",
  "versionStr":"PASOE 11.5.0",
  "operation":"GET OE_REST_TRANSPORT METRICS"
}

Get runtime metrics of a transport (APSV)

Description
Get the runtime metrics for the APSV transport for a Web application deployed in an ABL application.

HTTP Operation
GET

URI
//host_name:port/oemanager/applications/ABL_app_name/webapps/Web_app_name/transports/metrics

Media type
application/vnd.progress+json

Request body
NA
Response codes

200 Success
500 Unexpected Server Error

Command-line example

curl -X GET -v
http://localhost:16680/oemanager/applications/pas1/webapps/ROOT/transports/metrics

Response body example

{
  "operation":"GET OE_APSV_TRANSPORT METRICS",
  "result":{
    "sessionErrors":0,
    "sessionRequest":0,
    "disconnectRequest":0,
    "connectErrors":0,
    "connectRequest":0,
    "statusRequest":0,
    "disconnectErrors":0,
    "forbiddenErrors":0,
    "type":"OE_APSV_TRANSPORT",
    "accessTime":"2014-07-18T04:34:40.743-04:00",
    "lastResetTime":"2014-07-18T04:34:22.314-04:00",
    "versionNo":1,
    "versionStr":"PASOE 11.5.0",
    "outcome":"SUCCESS",
    "errmsg":"NA"
  }
}

Reset runtime metrics for a transport (SOAP/REST)

Description
Reset aggregate runtime metrics for a SOAP or REST transport of a Web application deployed in an ABL application.

HTTP Operation
DELETE

URI

//host_name:port/oemanager/applications/
ABL_app_name/webapps/Web_app_name/transports/{soap|rest}/metrics

Media type
application/vnd.progress+json

Request body
NA
Response codes

200 Success
500 Unexpected Server Error

Command-line example (SOAP)

curl -X DELETE -v

Response body example (SOAP)

```
{
  "result":"NA",
  "outcome":"SUCCESS",
  "versionNo":1,
  "errmsg":"NA",
  "versionStr":"PASOE 11.5.0",
  "operation":"RESET OE_REST_TRANSPORT METRICS"
}
```

Command-line example (REST)

curl -X DELETE -v

Response body example (REST)

```
{
  "result":"NA",
  "outcome":"SUCCESS",
  "versionNo":1,
  "errmsg":"NA",
  "versionStr":"PASOE 11.5.0",
  "operation":"RESET OE_SOAP_TRANSPORT METRICS"
}
```

Reset runtime metrics of a transport (APSV)

Description
Reset aggregate runtime metrics for an APSV transport of a Web application deployed in an ABL application.

HTTP Operation
DELETE

URI

```
//host_name:port/oemanager/applications/ABL_app_name
/webapps/Web_app_name/transports/metrics
```

Media type

application/vnd.progress+json
Request body
NA

Response codes
200 Success
500 Unexpected Server Error

Command-line example

curl -X DELETE -v
http://localhost:16680/oemanager/applications/pas1/webapps/ROOT/transports/metrics

Response body example

{  
"result":"NA",
"outcome":"SUCCESS",
"versionNo":1,
"errmsg":"NA",
"versionStr":"PASOE 11.5.0",
"operation":"RESET OE_APSV_TRANSPORT METRICS"
}

List deployed services (SOAP/REST)

Description
Get a list all the deployed SOAP or REST services in a Web application deployed in an ABL application.

HTTP Operation
GET

URI

//host_name:port/oemanager/applications/App_name/webapps/Web_app_name/transports/{soap | rest}/oeservices

Media type
application/vnd.progress+json

Response codes
200 Success
500 Unexpected Server Error

Request body
NA
Command-line example (SOAP)

curl -X GET -v

Response body example(SOAP)

{
    "result": {
        "OESoapServiceDesc": [
            {
                "archiveLocation": "/webapps/ROOT/WEB-INF/adapters/soap/ROOT/SimpleTest.wsdl",
                "archiveName": "SimpleTest.wsm",
                "accessUrl": "urn:nxgas:simpletestsf",
                "oetype": "SOAP_DESCRIPTOR",
                "version": "11.5.0",
                "description": "PAS OpenEdge SOAP Service Artifact",
                "name": "SimpleTest",
                "state": "ENABLED",
                "type": "OPENEDGE"
            }
        ],
        "operation": "GET SOAP TRANSPORT DESCRIPTORS",
        "versionStr": "PASOE 11.5.0",
        "versionNo": 1,
        "outcome": "SUCCESS",
        "errmsg": ""
    }
}

Command-line example (REST)

curl -X GET -v

Response body example(REST)

{
    "result": {
        "OERestServiceDesc": [
            {
                "archiveLocation": "/webapps/ROOT/WEB-INF/adapters/rest/_oepingService/_oepingService.paar",
                "archiveName": "_oepingService.paar",
                "accessUrl": "/rest/_oepingService",
                "oetype": "REST_DESCRIPTOR",
                "version": "11.5.0",
                "description": "PAS OpenEdge REST Service Descriptor",
                "name": "_oepingService",
                "type": "OPENEDGE",
                "uri": "http://localhost:16680/rest/_oepingService"
            }
        ],
        "operation": "GET REST TRANSPORT DESCRIPTORS",
        "versionStr": "PASOE 11.5.0",
        "versionNo": 1,
        "outcome": "SUCCESS",
        "errmsg": ""
    }
}
Get information about a service (SOAP/REST)

Description
Get details about a named SOAP or REST service in a Web application deployed in an ABL application.

HTTP Operation
GET

URI
//host_name:port/oemanager/applications/ABL_app_name
/webapps/Web_app_name/transports/{soap|rest}/oeservices/{SoapServiceName|RestServiceName}

Media type
application/vnd.progress+json

Response codes
200 Success
500 Unexpected Server Error

Request body
NA

Command-line example (SOAP)
curl -X GET -v

Response body example (SOAP)
{
   "result": {
      "OESoapServiceDesc": [
         {
            "archiveLocation": "/WEB-INF/adapters/soap/ROOT/",
            "archiveName": "SimpleTest.wsm",
            "accessUrl": "urn:nxgas:simpletestsf",
            "oetype": "SOAP_DESCRIPTOR",
            "version": "11.5.0",
            "description": "PAS OpenEdge SOAP Service Artifact",
            "name": "SimpleTest",
            "state": "ENABLED",
            "type": "OPENEDGE"
         }
      ],
      "operation": "GET SOAP TRANSPORT DESCRIPTOR",
      "versionStr": "PASOE 11.5.0",
      "versionNo": 1,
      "outcome": "SUCCESS",
      "errmsg": ""
   }
}
Command-line example (REST)

curl -X GET -v

Response body example (REST)

{
   "result": {
      "OERestServiceDesc": [
         {
            "archiveLocation": "/webapps/ROOT/WEB-INF/adapters/rest/_oepingService/",
            "archiveName": "_oepingService.paar",
            "accessUrl": "/rest/_oepingService",
            "oetype": "REST_DESCRIPTOR",
            "version": "11.5.0",
            "description": "PAS OpenEdge REST Service Descriptor",
            "name": "_oepingService",
            "type": "OPENEDGE",
            "uri": "http://localhost:16680/rest/_oepingService"
         }
      ],
      "operation": "GET REST TRANSPORT DESCRIPTOR",
      "versionStr": "PASOE 11.5.0",
      "versionNo": 1,
      "outcome": "SUCCESS",
      "errmsg": ""
   }
}

Get configuration properties of a service (SOAP)

Description
Get configuration properties of a named SOAP service in a Web application deployed in an ABL application.

HTTP Operation
GET

URI

//host_name:port/oemanager/applications/ABL_app_name/webapps/Web_app_name
/transports/soap/oeservices/SoapServiceName/properties

Media type
application/vnd.progress+json

Response body
NA

Response codes
200 Success
500 Unexpected Server Error
Command-line example

```
```

Request body example

```
{"result":
  
    
    "idleSessionTimeout":0,
    "waitIfBusy":0,
    "connectionLifetime":0,
    "requestWaitTimeout":-1,
    "staleO4GLObjectTimeout":0,
    "clientASKActivityTimeout":60,
    "initialSessions":1,
    "clientASKResponseTimeout":60,
    "minIdleConnections":0,
    "maxSessions":0,
    "minSessions":1,
    "appServerKeepalive":"denyClientASK,allowServerASK",
    "serviceFaultLevel":2

  },

  "outcome":"SUCCESS",
  "versionNo":1,
  "errmsg":"NA",
  "versionStr":
    "PASOE 11.5.0",
  "operation":"GET SOAP TRANSPORT PROPERTIES"
}
```

Deploy a new service (REST)

**Description**

Deploy a new REST service in a WebApp deployed in an ABL application.

**HTTP Operation**

POST

**URI**

//host_name:port/oemanager/applications/ABL_app_name/webapps
/Web_app_name/transports/rest/oeservices/

**Media type**

application/vnd.progress.paar+zip

**Content disposition**

attachment; filename={ AppNewService.paar | AppNewService.zip }
Response codes

200 Success
500 Unexpected Server Error

Command-line example

```
```

Request body example

application/vnd.progress.paar+zip

Response body example

```
{
 "result": [
  "OERestServiceDesc": [
   {
    "archiveLocation": "WEB-INF/adapters/rest/PingService1/PingService1.paar",
    "archiveName": "PingService1.paar",
    "accessUrl": "/rest/PingService1",
    "oetype": "REST_DESCRIPTOR",
    "version": "11.5.0",
    "description": "PAS OpenEdge REST Service Descriptor",
    "name": "PingService1",
    "type": "OPENEDGE",
    "uri": "http://localhost:16680/rest/PingService1"
   }
  ]
 },
 "operation": "DEPLOY REST TRANSPORT DESCRIPTOR",
 "versionStr": "PASOE 11.5.0",
 "versionNo": 1,
 "outcome": "SUCCESS",
 "errmsg"
}
```

Deploy a new service (SOAP)

Description
Deploy a new SOAP service in a WebApp deployed in an ABL application.

HTTP Operation
POST

URI

```
//host_name:port/oemanager/applications/ABL_app_name/webapps/Web_app_name/transports/soap/oeservices/
```
Media type

application/vnd.progress.wsm+xml

Content disposition

attachment; filename=AppNewService.wsm

Response codes

200 Success
500 Unexpected Server Error

Command-line example

curl -X POST -v
-data-binary @SimpleTest2.wsm
-H "Accept: application/vnd.progress+json"
-H "Content-Type: application/vnd.progress.wsm+xml"
-H "Content-Disposition: attachment; filename=SimpleTest2.wsm"

Request body example

application/vnd.progress.wsm+xml

Response body example

{
  "result": {
    "OESoapServiceDesc": [
      {
        "archiveLocation": "WEB-INF/adapters/soap/ROOT/SimpleTest.wsdl",
        "archiveName": "SimpleTest.wsm",
        "accessUrl": "urn:nxgas:simpletestsf",
        "oetype": "SOAP_DESCRIPTOR",
        "version": "11.5.0",
        "description": "PAS OpenEdge SOAP Service Artifact",
        "name": "SimpleTest",
        "state": "ENABLED",
        "type": "OPENEDGE"
      }
    ],
    "operation": "DEPLOY SOAP TRANSPORT DESCRIPTOR",
    "versionStr": "PASOE 11.5.0",
    "versionNo": 1,
    "outcome": "SUCCESS",
    "errmsg": ""
  }
}

Update runtime properties of a service (SOAP)

Description

Update the runtime properties of a SOAP service in the Web application of a deployed ABL application.
HTTP Operation

PUT

URI

//host_name:port/oemanager/applications/ABL_app_name
/webapps/Web_app_name/transports/soap/oeservices/SoapServiceName/properties

Media type

application/vnd.progress+json

Response codes

200 Success
500 Unexpected Server Error

Command-line example

curl -X PUT -v
http://oelxdev03.bedford.progress.com:16680/oemanager/server/oeservices/ROOT/transports/soap/service/TestService2/properties
-H "Content-Type: application/vnd.progress+json" -d '{"maxSessions":1}'

Request body example

{"maxSessions":1}

Response body example

{"result":
{"idleSessionTimeout":0,
 "waitIfBusy":1,
 "connectionLifetime":0,
 "requestWaitTimeout":-1,
 "staleO4GLObjectTimeout":0,
 "clientASKActivityTimeout":60,
 "initialSessions":1,
 "clientASKResponseTimeout":60,
 "minIdleConnections":0,
 "maxSessions":1,
 "minSessions":1,
 "appServerKeepalive":"denyClientASK,allowServerASK",
 "serviceFaultLevel":2
 },
 "outcome":"SUCCESS",
 "versionNo":1,
 "errmsg":"NA",
 "versionStr":
 "PASOE 11.5.0",
 "operation":"SET SOAP TRANSPORT PROPERTIES"}

Get runtime metrics for a service (SOAP)

Description

Update the runtime metrics of a SOAP service in the Web application deployed in an ABL application.
HTTP Operation

GET

URI

//host_name:port/oemanager/applications/ABL_app_name
/webapps/Web_app_name/transports/soap/oeservices/SoapServiceName/metrics

Media type

application/vnd.progress+json

Request body

NA

Response codes

200 Success
500 Unexpected Server Error

Command-line example

curl -X GET -v

Response body example

{"result":
"requests":0,
"appserverConnections":0,
"nameserverError":0,
"activeRequests":0,
"providerError":0,
"subAppObjects":0,
"objPoolFull":0,
"objPoolExpired":0,
"connectError":0,
"faults":0,
"ABLClientError":0,
"procObjects":0,
"startTime":"2014-07-17T08:40:07.315-04:00",
"appObjects":0,
"objNotFound":0,
"ABLServerError":0,
"accessTime":"2014-07-18T02:27:07.212-04:00",
"ABLAppError":0,
"clientError":0,
"type":"OE_SOAP_TRANSPORT_APP"
},
"outcome":"SUCCESS",
"versionNo":1,
"errmsg":"NA",
"versionStr":"PASOE 11.5.0",
"operation":"GET OE_SOAP_TRANSPORT_APP METRICS"}
Reset runtime metrics for a service (SOAP)

Description
Reset runtime metrics for a SOAP service in a Web application deployed in an ABL application.

HTTP Operation
DELETE

URI
//host_name:port/oemanager/applications/ABL_app_name/webapps/Web_app_name/transports/soap/oeservices/SoapServiceName/metrics

Media type
application/vnd.progress+json

Request body
NA

Response codes
200 Success
500 Unexpected Server Error

Command-line example

Response body example
{
    "result":"NA",
    "outcome":"SUCCESS",
    "versionNo":1,
    "errmsg":"NA",
    "versionStr":"PASOE 11.5.0",
    "operation":"RESET OE_SOAP_TRANSPORT_APP METRICS"
}

Enable or disable a service (SOAP)

Description
Enable or disable a SOAP service in a Web application deployed in an ABL application.

HTTP Operation
PUT
URI

//host_name:port/oemanager/applications/ABL_app_name/webapps/W eb_app_name/transports/soap/oeservices/SoapServiceName/state

Media type

application/vnd.progress+json

Request body

{
   "STATE" : "{ ENABLED | DISABLED }"
}

Response codes

200 Success
500 Unexpected Server Error

Command-line example


Response body example

{"state":"DISABLED"}

Remove a service (SOAP/REST)

Description
Remove a specified SOAP or REST service from a Web application deployed in an ABL application.

HTTP Operation
DELETE

URI

//host_name:port/oemanager/applications/ABL_app_name/webapps/WEB_app_name/transports/soap | rest/oeservices\{SoapServiceName | RestServiceName\}

Media type

application/vnd.progress+json

Response codes

500 Unexpected Server Error
Request body
NA

Command-line example (SOAP)

curl -X DELETE -v

Response body example (SOAP)

```json
{
   "result": {
      "OESoapServiceDesc": [
      {
         "status": "INITIALIZED",
         "descriptors": [],
         "oetype": "SOAP",
         "version": "11.4ALPHA",
         "description": "PAS OpenEdge SOAP Service.",
         "name": "SOAP",
         "state": "ENABLED",
         "type": "OPENEDGE",
         "uri": "http://localhost.com:16680/soap"
      }
      ],
   "operation": "UNDEPLOY SOAP TRANSPORT DESCRIPTOR",
   "versionStr": "PASOE 11.5.0",
   "versionNo": 1,
   "outcome": "SUCCESS",
   "errmsg": ""
   }
}
```

Command-line example (REST)

curl -X DELETE -v

Response body example (REST)

```json
{"result": {
   "OERestServiceDesc": [
   {
      "status": "INITIALIZED",
      "contextPath": "/\wrk\pas1\webapps\ROOT",
      "descriptors": [],
      "archiveLocation": "WEB-INF\adapters\rest\_/oepingService\_/oepingService.paar",
      "archiveName": ",oepingService.paar",
      "accessUrl": "/rest\_/oepingService",
      "oetype": "REST_DESCRIPTOR",
      "version": "11.5.0",
      "description": "PAS OpenEdge REST Service Descriptor",
      "name": ",oepingService",
      "type": "OPENEDGE",
      "uri": "http://localhost.com:16680/rest\_/oepingService"
   }
   ]
}
```

Chapter 11: REST API Reference for oemanager.war
Application management

This section describes the REST services that are available for managing ABL and Web applications deployed on the Progress Application Server.

See also
List ABL application information on page 203
Get information about a specific ABL application on page 205
List Web applications deployed to an ABL application on page 207
Get information about a specific Web application on page 209

List ABL application information

Description
Get information about ABL applications from the OEManager.

HTTP Operation
GET

URI
//host_name:port/oemanager/applications/

Media type
application/vnd.progress+json

Request body
NA

Response codes
200 Success
500 Unexpected Server Error
Command-line example

curl -X GET -v http://localhost:16680/oemanager/applications/

Response body example

"result": {
  "Application": [
    "webapps": [
      {
        "applicationName": "pasl",
        "transports": [
          {
            "status": "INITIALIZED",
            "descriptors": [
              {
                "archiveLocation": "/wrk/pas1/webapps/ROOT/WEB-INF/adapters/soap//ROOT/SimpleTest.wsdl",
                "archiveName": "SimpleTest.wsm",
                "accessUrl": "urn:nxgas:simpletestsf",
                "oetype": "SOAP_DESCRIPTOR",
                "version": "11.5.0",
                "description": "PAS OpenEdge SOAP Service Artifact",
                "name": "SimpleTest",
                "state": "DISABLED",
                "type": "OPENEDGE"
              }
            ],
            "oetype": "SOAP",
            "version": "11.4ALPHA",
            "description": "PAS OpenEdge SOAP Service.",
            "name": "SOAP",
            "state": "DISABLED",
            "type": "OPENEDGE",
            "uri": "http://localhost:16680/soap"
          },
          {
            "status": "INITIALIZED",
            "oetype": "APSV",
            "version": "v115.PASOE (11-SEPT-2014)",
            "description": "PAS OpenEdge APSV Transport.",
            "name": "APSV",
            "state": "ENABLED",
            "type": "OPENEDGE",
            "uri": "http://localhost:16680/apsv"
          },
          {
            "status": "INITIALIZED",
            "contextPath": "//wrk/pas1/webapps/ROOT",
            "descriptors": [
              {
                "archiveLocation": "/wrk/pas1/webapps/ROOT/WEB-INF/adapters/rest//ROOT/_oepingService/_oepingService.paar",
                "archiveName": "_oemingService.paar",
                "accessUrl": "/rest/_oepingService",
                "oetype": "REST_DESCRIPTOR",
                "version": "11.5.0",
                "description": "PAS OpenEdge REST Service Descriptor",
                "name": "_oepingService",
                "type": "OPENEDGE",
                "uri": "http://localhost:16680/rest/_oepingService"
              }
            ],
            "oetype": "REST",
            "version": "v11.4.0 (12-Dec-2013)",
            "description": "PAS OpenEdge REST Transport.";
```
Get information about a specific ABL application

**Description**

Get information about a specified application from the OEmanager.

**HTTP Operation**

GET

**URI**

//host_name:port/oemanager/applications/ABL_app_name

**Media type**

application/vnd.progress+json

**Request body example**

NA

**Response codes**

200 Success
500 Unexpected Server Error
Command-line example

curl -X GET -v http://localhost:16680/oemanager/applications/pas1

Response body example

```json
{
  "result": {
    "Application": [
      {
        "webapps": [
          {
            "applicationName": "pas1",
            "transports": [
              {
                "status": "INITIALIZED",
                "descriptors": [
                  {
                    "archiveLocation": "/wrk/pas1/webapps/ROOT/WEB-INF/adapters/soap//ROOT/SimpleTest.wsdl",
                    "archiveName": "SimpleTest.wsm",
                    "accessUrl": "urn:nxgas:simplestestsf",
                    "oetype": "SOAP_DESCRIPTOR",
                    "version": "11.5.0",
                    "description": "PAS OpenEdge SOAP Service Artifact",
                    "name": "SimpleTest",
                    "state": "DISABLED",
                    "type": "OPENEDGE"
                  }
                ],
                "oetype": "SOAP",
                "version": "11.4ALPHA",
                "description": "PAS OpenEdge SOAP Service.",
                "name": "SOAP",
                "state": "DISABLED",
                "type": "OPENEDGE",
                "uri": "http://localhost:16680/soap"
              },
              {
                "status": "INITIALIZED",
                "oetype": "APSV",
                "version": "v115.PASOE (11-SEPT-2014)",
                "description": "PAS OpenEdge APSV Transport.",
                "name": "APSV",
                "state": "ENABLED",
                "type": "OPENEDGE",
                "uri": "http://localhost:16680/apsv"
              },
              {
                "status": "INITIALIZED",
                "contextPath": "/wrk/pas1/webapps/ROOT",
                "descriptors": [
                  {
                    "archiveLocation": "/wrk/pas1/webapps/ROOT/WEB-INF/adapters/soap//ROOT/_oepingService/_oepingService.paar",
                    "archiveName": "_oepingService.paar",
                    "accessUrl": "/rest/_oepingService",
                    "oetype": "REST_DESCRIPTOR",
                    "version": "11.5.0",
                    "description": "PAS OpenEdge REST Service Descriptor",
                    "name": "_oepingService",
                    "type": "OPENEDGE",
                    "uri": "http://localhost:16680/rest/_oepingService"
                  }
                ],
                "oetype": "REST",
                "version": "v11.4.0 (12-Dec-2013)",
                "description": "PAS OpenEdge REST Transport."
              }
            ]
          }
        ]
      }
    ]
  }
}```
List Web applications deployed to an ABL application

Description
Get information from the OERManager about all the Web applications deployed to an ABL application.

HTTP Operation
GET

URI
//host_name:port/oemanager/applications/ABL_app_name/webapps

Media type
application/vnd.progress+json

Request body
NA

Response codes
200 Success
500 Unexpected Server Error
Command-line example

curl -X GET -v http://localhost:16680/oemanager/applications/pas1/webapps

Response body example

```json
{
  "result": [
    {
      "WebApp": [
        {
          "applicationName": "pas1",
          "transports": [
            {
              "status": "INITIALIZED",
              "descriptors": [
                {
                  "archiveLocation": "/wrk/pas1/webapps/ROOT/WEB-INF/adapters/soap/ROOT/SimpleTest.wsdl",
                  "archiveName": "SimpleTest.wsm",
                  "accessUrl": "urn:nxgas:simpletestsf",
                  "oetype": "SOAP_DESCRIPTOR",
                  "version": "11.5.0",
                  "description": "PAS OpenEdge SOAP Service Artifact",
                  "name": "SimpleTest",
                  "state": "DISABLED",
                  "type": "OPENEDGE"
                }]
              }],
            {
              "status": "INITIALIZED",
              "transports": [
                {
                  "oetype": "SOAP",
                  "version": "11.4ALPHA",
                  "description": "PAS OpenEdge SOAP Service.",
                  "name": "SOAP",
                  "state": "DISABLED",
                  "type": "OPENEDGE",
                  "uri": "http://localhost:16680/soap"
                }
              ],
            }],
            {
              "status": "INITIALIZED",
              "transports": [
                {
                  "contextPath": "/wrk/pas1/webapps/ROOT",
                  "descriptors": [
                    {
                      "archiveLocation": "WEB-INF/adapters/rest/_oepingService/_oepingService.paar",
                      "archiveName": "_oepingService.paar",
                      "accessUrl": "/rest/_oepingService",
                      "oetype": "REST_DESCRIPTOR",
                      "version": "11.5.0",
                      "description": "PAS OpenEdge REST Service Descriptor",
                      "name": "_oepingService",
                      "type": "OPENEDGE",
                      "uri": "localhost:16680/rest/_oepingService"
                    }
                  ]
                }
              ],
              "oetype": "REST",
              "version": "v11.4.0 (12-Dec-2013)",
              "description": "PAS OpenEdge REST Transport.",
              "name": "REST",
              "state": "ENABLED"
```
Get information about a specific Web application

**Description**
Get information from the OEManager about a specific Web application deployed to an ABL application.

**HTTP Operation**
GET

**URI**

//host_name:port/oemanager/applications/ABL_app_name/webapps/Web_app_name

**Media type**
application/vnd.progress+json

**Request body**
NA

**Response codes**

200 Success
500 Unexpected Server Error

**Command-line example**

curl -X GET -v http://localhost:16680/oemanager/applications/pas1/webapps/ROOT

**Response body example**

{
    "result": {
        "WebApp": [
Chapter 11: REST API Reference for oemanager.war

{  
  "applicationName": "pas1",
  "transports": [
    {
      "status": "INITIALIZED",
      "descriptors": [
        {
          "archiveLocation": "/wrk/pas1/webapps/ROOT/WEB-INF/adapters/soap//ROOT/SimpleTest.wsdl",
          "archiveName": "SimpleTest.wsm",
          "accessUrl": "urn:nxgas:simpletestsf",
          "oetype": "SOAP_DESCRIPTOR",
          "version": "11.5.0",
          "description": "PAS OpenEdge SOAP Service Artifact",
          "name": "SimpleTest",
          "state": "DISABLED",
          "type": "OPENEDGE"
        }
      ],
      "oetype": "SOAP",
      "version": "11.4ALPHA",
      "description": "PAS OpenEdge SOAP Service.",
      "name": "SOAP",
      "state": "DISABLED",
      "type": "OPENEDGE",
      "uri": "http://localhost:16680/soap"
    },
    {
      "status": "INITIALIZED",
      "oetype": "APSV",
      "version": "v115.PASOE (11-SEPT-2014)",
      "description": "PAS OpenEdge APSV Transport.",
      "name": "APSV",
      "state": "ENABLED",
      "type": "OPENEDGE",
      "uri": "http://localhost:16680/apsv"
    },
    {
      "status": "INITIALIZED",
      "contextPath": "/wrk/pas1/webapps/ROOT",
      "descriptors": [
        {
          "archiveLocation": "WEB-INF/adapters/rest/_oepingService/_oepingService.paar",
          "archiveName": "_oepingService.paar",
          "accessUrl": "/rest/_oepingService",
          "oetype": "REST_DESCRIPTOR",
          "version": "11.5.0",
          "description": "PAS OpenEdge REST Service Descriptor",
          "name": "_oepingService",
          "type": "OPENEDGE",
          "uri": "http://localhost:16680/rest/_oepingService"
        }
      ],
      "oetype": "REST",
      "version": "v11.4.0 (12-Dec-2013)",
      "description": "PAS OpenEdge REST Transport.",
      "name": "REST",
      "state": "ENABLED",
      "type": "OPENEDGE",
      "uri": "http://localhost:16680/rest"
    }
  ],
  "oetype": "WEBAPP",
  "version": "11.5.0",
  "description": "PAS OpenEdge WebApp",
  "name": "ROOT",
  "state": "STARTED",
  "type": "OPENEDGE",
}
"uri": "http://localhost:16680/"
}
],
"operation": "GET OEABL SERVICES",
"versionStr": "PASOE 11.5.0",
"versionNo": 1,
"outcome": "SUCCESS",
"errmsg": ""
}
Index

A
ABL application information 203, 205, 207, 209
ABL applications 64
Administrative tools 35
Agent management 170
Agent status 171
Amazon load balancing 77
Apache load balancing 77
Apache proxy host load balancing 69
Apache proxy load balancing 70–71
Application management 203
Audience 11
Auto scaling 77

C
Cancel a request 168
clean action 139
Clients 161
Clusters 67
config action 135
Configuration files 24
Configuration properties 163, 165, 178–179
Connection information 173
CORS support 107
create 132
Creating instances 56

D
delete 134
deploy 123
Deployed services 191
deployREST.sh 62
deploySOAP.sh 63
Development server 79
Development servers 20
Directory structure extensions 33
disable 129

E
Elastic load balancing 77
enable 128
Enabling Spring Security 93
ev 156

F
feature action 137

H
help 155
HTTP sessions 65
HTTPS 80

I
info 122
instances 140
Instances
creating 56
managing 55
Running as a UNIX daemon 58
Running as a Windows service 58

J
JConsole 49
JMX 49
JVM security manager 87

L
LDAP security configurations 96
leaks 127
list 121
List agents 170
List clients 161
List sessions 160
Load balancing
 Apache proxy host 69
 Apache proxy load balancing 70
 Tomcat load balancing 73
Logging 23
login models 106

M
Managing instances 55
Managing Web applications 61
MBeans 49
MemoryRealm 88

O
OEPROP 45
OERealm 97
OpenEdge Explorer 48
OpenEdge Management 48
Organization 12

P
PAS for OpenEdge
  development server 79
  production server 79
PASMAN 43
PASOE
  administrative utilities 35
  pasoestart 37
  plist 141
Production server 79
Production servers 20
Progress Application Server
  directory structure extensions 33
  extensions to Apache Tomcat 19
Progress Application Server for OpenEdge
  load balancing 69
Properties
  PAS properties 25
  Properties files 24
  Proxy load balancing 72
Purpose 11

R
Realms 87
  register 142
  reload 125
Remote access filters 89
Request information 162, 174
Reset runtime metrics 167
resources 130
REST API for omanager.war
  cancel a request 168
  clients 161
  deploy a new service 195–196
  enable or disable a service 200
  enable or disable a transport 185–186
  get ABL application information 203, 205, 209
  get agent status 171
  get configuration properties 163, 178
  get connection information 173
  get request information 162, 174
  get runtime metrics 166
  get service configuration properties 194
  get service information 193
  get service runtime metrics 198
  get session metrics 172
  get transport configuration properties 182
  get transport metrics 187–188
  list agents 170
  list deployed services 191
  list Web applications 207
  removing a service 201

REST API for omanager.war (continued)
  reset runtime metrics 167
  reset service runtime metrics 200
  reset transport metrics 189–190
  sessions 160
  stop an agent 180
  terminate a session 169
  update configuration properties 165, 179
  update service runtime properties 197
  update transport properties 183
REST API Reference 159
REST service deployment (if needed)
  Nest index terms for sub entries 62
Roles
  assigning to users 88
ROOT Web application 20
Runtime metrics 166

S
Security 79
  service 144
Service deployment 195–196
Service enable or disable 200
Service information 193–194
Service removal 201
Service runtime metrics 198, 200
Service runtime properties 197
Session management 159
Session metrics 172
  sessions 131
  Sessions 160
  showproc 146
SOAP service deployment 63
Spring Security 87
SSO support 108
start 147
status 126
Sticky sessions 67
stop 148
Stop an agent 180

T
tcman 116
TCMAN
  clean 139
  config 135
  create 132
  customizing 118
  delete 134
  deploy 123
  disable 129
  enable 128
  env 156
  extending 118
TCMAN (continued)
  feature 137
  general actions 155
  help 155
  info 122
  instances 140
  leaks 127
  list 121, 141
  manager actions 120
  register 142
  reload 125
  resources 130
  server actions 132
  service 144
  sessions 131
  showproc 146
  start 147
  status 126
  stop 148
  test 150
  undeploy 124
  unregister 151
  using 115
  version 149
  worker 152
Terminate a session 169
  test 150
  Tomcat load balancing 73–75
  Transport management
    URLs 20
  Transport metrics 187–190

Transport properties 182–183
  Transports 185–186

U
  undeploy 124
  UNIX daemon 58
  unregister 151
  URLs
    WEB transport 21

V
  version 149

W
  Web application deployment 62
  Web application security configurations 91
  Web applications
    managing 61
    Packaging 63
  WEB transport URL 21
  WebSpeed
    configuration 111
    configuring openedge.properties 112
    management 111
    managing application files 113
    security 113
  Windows service 58
  worker 152