Pacific™ Application Server for OpenEdge®: Administration Guide
Notices

© 2015 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, EasyX, Fathom, Icenium, Kendo UI, Making Software Work Together, OpenEdge, Powered by Progress, Progress, Progress Control Tower, Progress RPM, Progress Software Business Making Progress, Progress Software Developers Network, Rollbase, RulesCloud, RulesWorld, 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, AppsAlive, AppServer, BravePoint, BusinessEdge, DataDirect Spy, DataDirect SupportLink, Future Proof, High Performance Integration, Modulus, NativeScript, OpenAccess, Pacific, ProDataSet, Progress Arcade, Progress Pacific, Progress Profiles, Progress Results, Progress RFID, Progress 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:

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).

October 2015

Last updated with new content: Release 11.6.0
# Table of Contents

Preface...........................................................................................................11  
Purpose...........................................................................................................11  
Audience...........................................................................................................11  
Organization...................................................................................................12  
Using this manual...........................................................................................12  
  References to ABL compiler and run-time features...........................................13  
  References to ABL data types..........................................................................13  
Typographical conventions..............................................................................13  
Examples of syntax descriptions.....................................................................15  
  Long syntax descriptions split across lines......................................................16  
  Complex syntax descriptions with both required and optional elements...........17  
OpenEdge messages.........................................................................................17  
  Obtaining more information about OpenEdge messages.................................18  

Chapter 1: Overview of PAS for OpenEdge administration......................19  
Administrative utilities....................................................................................20  
  TCMAN...........................................................................................................20  
  OEPROP.......................................................................................................21  
OpenEdge Management and OpenEdge Explorer............................................24  
JMX and JConsole............................................................................................24  
Administrative scripts.....................................................................................24  
Development servers compared to production servers....................................26  
Managing the ROOT Web application..............................................................27  
Transport URLs..............................................................................................27  
  Modifying the WEB transport URL.................................................................28  
Tomcat logging................................................................................................29  
Configuration and properties files.................................................................30  
Pacific Application Server properties...............................................................31  
Pacific Application Server directories.............................................................40  

Chapter 2: Working with Instances............................................................41  
Overview of instances in PAS for OpenEdge ..................................................42  
Instance management with TCMAN.................................................................43  
Creating instances with TCMAN.....................................................................44  
Installing and running an instance as a UNIX daemon......................................46  
Installing and running an instance as a Windows service..................................47
Local configurations..............................................................................................................................84
Adding new users in local security configurations.................................................................84
Using Jacksum to generate hashed and salted passwords.........................................................85
Access control in local security configurations.........................................................................86

SPA security configurations...................................................................................................................87
Creating an OERealm service interface.....................................................................................88
Updating the SPA configuration file............................................................................................90
SPA security considerations.......................................................................................................95

LDAP security configurations................................................................................................................97

SAML security configurations................................................................................................................98
Creating and configuring metadata files.....................................................................................99
Registering the metadata file with an Identity Provider............................................................100
Updating the SAML configuration file.......................................................................................101

Container security configurations........................................................................................................105
CORS support.....................................................................................................................................105
Adding CORS support..............................................................................................................106
OECORSFilter properties.........................................................................................................109

Single sign-on (SSO) support in PAS for OpenEdge .................................................................111
Configuring the ClientPrincipal.................................................................................................111
Enabling SSO for Web applications............................................................................................113

Chapter 8: WebSpeed configuration and management..................................................117
Tools for managing and configuring WebSpeed on PAS for OpenEdge.................................117
Configuring openedge.properties.................................................................................................119
Security...............................................................................................................................................119
Migrating classic WebSpeed applications..................................................................................120

Chapter 9: TCMAN Reference.................................................................................................123
Using TCMAN......................................................................................................................................123
The tcman command.........................................................................................................................124
Extending TCMAN...............................................................................................................................126
Manager actions.................................................................................................................................129
List deployed applications (list)... .................................................................................................129
Display OS and server information (info)... .................................................................................130
Deploy a Web application (deploy)... ..............................................................................................131
Undeploy a Web application (undeploy)... ..................................................................................132
Reload a Web application (reload)... .............................................................................................133
Display detailed server status (status)... ....................................................................................134
Display memory leaks (leaks)... .....................................................................................................136
Start a Web application (enable)... .................................................................................................136
Stop a Web application (disable)... ...............................................................................................137
Display global server resources (resources)... .............................................................................138
Display Web application HTTP sessions (sessions)... ..................................................................139
Chapter 10: REST API Reference for oemanager.war

Chapter 10: REST API Reference for oemanager.war

Session management.................................................................................................................................168
Get session information..........................................................................................................................168
Get client information............................................................................................................................169
Get current request information........................................................................................................171
Get configuration properties..................................................................................................................172
Update configuration properties........................................................................................................173
Get runtime metrics...............................................................................................................................174
Reset runtime metrics...........................................................................................................................175
Cancel a request........................................................................................................................................176
Terminate a session..................................................................................................................................177
Agent management...................................................................................................................................178
List agents................................................................................................................................................178
Get agent status information..................................................................................................................179
Get session metrics................................................................................................................................180
Get connection information.....................................................................................................................181
Get request information..........................................................................................................................183
Get configuration properties...................................................................................................................187
Update configuration properties............................................................................................................188
Stop an agent............................................................................................................................................189
Transport management............................................................................................................................190
Get transport configuration properties (SOAP/REST)............................................................................190
Update properties of a transport (SOAP/REST)......................................................................................192
Preface

For details, see the following topics:

- Purpose
- Audience
- Organization
- Using this manual
- Typographical conventions
- Examples of syntax descriptions
- OpenEdge messages

Purpose

This is a guide to administering Pacific™ 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: Pacific 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 Pacific Application Server for OpenEdge.
Organization

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

Working with Instances on page 41
Explains how to create and manage instances of the core PAS for OpenEdge server.

Application deployment and management on page 49
Explains how to deploy, manage, and package PAS for OpenEdge Web applications.

HTTP sessions on page 53
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 57
Describes three alternatives for configuring load balancing among PAS for OpenEdge instances.

Security overview on page 69
Provides an overview of security features and concepts in PAS for OpenEdge.

Web application security configurations on page 77
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 117
Describes configuration of PAS for OpenEdge for deployment of WebSpeed applications.

TCMAN Reference on page 123
A command reference for the TCMAN command line management utility.

REST API Reference for oemanager.war on page 167
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 this manual

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 manual uses the following typographical and syntax conventions:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bold</strong></td>
<td>Bold typeface indicates commands or characters the user types, provides emphasis, or the names of user interface elements.</td>
</tr>
<tr>
<td>Convention</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><em>Italic</em></td>
<td>Italic typeface indicates the title of a document, or signifies new terms.</td>
</tr>
<tr>
<td>SMALL, BOLD CAPITAL LETTERS</td>
<td>Small, bold capital letters indicate OpenEdge key functions and generic keyboard keys; for example, GET and CTRL.</td>
</tr>
<tr>
<td>KEY1+KEY2</td>
<td>A plus sign between key names indicates a simultaneous key sequence: you press and hold down the first key while pressing the second key. For example, CTRL+X.</td>
</tr>
<tr>
<td>KEY1 KEY2</td>
<td>A space between key names indicates a sequential key sequence: you press and release the first key, then press another key. For example, ESCAPE H.</td>
</tr>
<tr>
<td><strong>Syntax:</strong></td>
<td></td>
</tr>
<tr>
<td>Fixed width</td>
<td>A fixed-width font is used in syntax, code examples, system output, and file names.</td>
</tr>
<tr>
<td>Fixed-width italics</td>
<td>Fixed-width italics indicate variables in syntax.</td>
</tr>
<tr>
<td><strong>Fixed-width bold</strong></td>
<td>Fixed-width bold italic indicates variables in syntax with special emphasis.</td>
</tr>
<tr>
<td>UPPERCASE fixed width</td>
<td>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.</td>
</tr>
<tr>
<td>Period (.) or colon (:)</td>
<td>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.</td>
</tr>
<tr>
<td>[ ]</td>
<td>Large brackets indicate the items within them are optional.</td>
</tr>
<tr>
<td>[]</td>
<td>Small brackets are part of ABL.</td>
</tr>
<tr>
<td>{ }</td>
<td>Large braces indicate the items within them are required. They are used to simplify complex syntax diagrams.</td>
</tr>
<tr>
<td>{}</td>
<td>Small braces are part of ABL. For example, a called external procedure must use braces when referencing arguments passed by a calling procedure.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>Ellipses indicate repetition: you can choose one or more of the preceding items.</td>
</tr>
</tbody>
</table>
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:

**Syntax**

```
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 }`:

**Syntax**

```
{ 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:

**Syntax**

```
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

```
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:

   ```bash
   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:

- Administrative utilities
- Development servers compared to production servers
- Managing the ROOT Web application
- Transport URLs
- Tomcat logging
- Configuration and properties files
- Pacific Application Server properties
- Pacific Application Server directories
Administrative utilities

To implement the Pacific Application Server, Progress-specific extensions were added to the standard administrative scripts and configuration files of Apache Tomcat. In addition, Apache Tomcat was extended to include new scripts, configuration files, and directories.

These extensions to the standard Apache Tomcat server were added in order to:

• Automate the setting of environment variables.
• Automate the configuration of the Pacific Application Server for specific Progress products (the OpenEdge ABL Service, for example).
• Re-configure Tomcat be primarily a production server rather than a development server. (An advantage of the Pacific Application Server is that it simplifies deployment and administration.)

Note:

Do not modify any of the scripts or configuration files in the installed instance of the Pacific Application Server located in $CATALINA_HOME. It is safer to create a new instance (using $CATALINA_HOME/bin/tcman.sh create). Edit files in the new instance, which is located in $CATALINA_BASE. You will preserve the original files in case you need to restore anything.

Also note that the Windows versions of these scripts have the same file name, but have a .bat extension.

TCMAN

the Pacific Application Server implements the TCMAN command line utility as an interface to administrative functionality provided by core Tomcat scripts. 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.

Syntax

The general for the TCMAN command is:

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

Parameters

$CATALINA_HOME|$CATALINA_BASE

Specify whether to run TCMAN from the root directory of the installed Pacific Application Server ($CATALINA_HOME) or from the root directory of an instance ($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 Pacific 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.

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

**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 123

## OEPROP

### Purpose

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

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

### Syntax

```plaintext
{${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:

```plaintext
[AppServer.SessMgr]
  agentExecFile=${psc.as.oe.dlc}/bin/_mproapsv
  agentListenerTimeout=300000
  agentLogEntryTypes=
  agentLogFile=
  agentLoggingLevel=2
  agentLogFile=0
  agentNumLogFiles=3
  agentStartupParam=-T ${catalina.base}/temp
  connectionWaitTimeout=3000
  idleAgentTimeout=300000
  idleConnectionTimeout=300000
  idleResourceTimeout=0
  idleSessionTimeout=3000000
  ipver=IPv4
  maxABLSessionsPerAgent=200
  maxAgents=10
  maxConnectionsPerAgent=16
  numInitialAgents=1
  publishDir=${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):**

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

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

  ```plaintext
  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`:**

  ```plaintext
  proenv> bin\oeprop.bat -f NewWebAppProperties.txt
  ```
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.

JMX and JConsole

The Pacific 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. 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.

JConsole, which is an application included with Java, can be used to monitor and manage MBeans. JConsole can be used for development and debugging, but you should not use JConsole on a local production server because JConsole itself uses significant resources. Using JConsole with a remote connection avoids this problem, although this usage requires more attention to security.

Administrative scripts

This topic is a brief overview of some of the administrative utilities for the Pacific Application Server, which are implemented as scripts in the Tomcat server's /bin directory. Each script has a UNIX (.sh extension) and a Windows (.bat extension) version.

Notice that the Pacific Application Server implements the TCMAN command line utility as an interface to administrative functionality provided by core Tomcat scripts. 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.

The following table lists the standard Tomcat utilities in $CATALINA_HOME/bin that were tailored by Progress to set the value of CATALINA_HOME and CATALINA_BASE to values appropriate for the Pacific Application Server and its instances.

<table>
<thead>
<tr>
<th>Script</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>startup.sh(.bat)</td>
<td>Initiates the start up of the Tomcat server and its web applications.</td>
</tr>
<tr>
<td></td>
<td>Functionality of this utility is also available through tcman.sh start.</td>
</tr>
<tr>
<td>shutdown.sh(.bat)</td>
<td>Initiates the Tomcat server's shutdown process, beginning with a graceful termination of any running web applications.</td>
</tr>
<tr>
<td></td>
<td>Functionality of this utility is also available through tcman.sh stop.</td>
</tr>
</tbody>
</table>
A standard Tomcat script that displays version and system information.
Functionality of this utility is also available through `tcman.sh info`

Displays the Tomcat server's configuration and environment information.
Functionality of this utility is also available through `tcman.sh test`

In addition, there are a number of supporting scripts that are called by Tomcat administrative utilities. Some of these supporting scripts are described in the following table.

**Table 2: Supporting administrative scripts for Pacific Application Server**

<table>
<thead>
<tr>
<th>Script</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>version.sh(.bat)</strong></td>
<td>A standard Tomcat script that displays version and system information. Functionality of this utility is also available through <code>tcman.sh info</code></td>
</tr>
<tr>
<td><strong>configtest.sh(.bat)</strong></td>
<td>Displays the Tomcat server's configuration and environment information. Functionality of this utility is also available through <code>tcman.sh test</code></td>
</tr>
<tr>
<td><strong>catalina.sh(.bat)</strong></td>
<td>Called by administrative utilities on startup or shutdown of the Tomcat server. Calls <code>setenv.sh(.bat)</code> (if it exists) and <code>setclasspath.sh(.bat)</code> to set environment.</td>
</tr>
<tr>
<td><strong>setclasspath.sh(.bat)</strong></td>
<td>Called by <code>catalina.sh(.bat)</code> to set JAVA_HOME or JRE_HOME if not already set. If they are set, it validates to ensure that the values are consistent with startup options.</td>
</tr>
</tbody>
</table>
| **setenv.sh(.bat)** | An optional file called by `catalina.sh(.bat)` on startup. Deploying this file is the standard way to customize the server environment for each running instance. `setenv.sh` performs the following operations:  
1. Set JVM memory size and the CATALINA environment variables common to all Progress product Web applications.  
2. Load Java system properties from `conf/server.xml`.  
3. Find and run any `progress_product_setenv.sh(.bat)` files. |

In addition to the standard Tomcat administrative utilities and supporting scripts described above, Pacific Application Server The following table lists the utility scripts in the Pacific Application Server /bin directory that were added by Progress to the standard Tomcat server.
Table 3: Pacific Application Server scripts

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tcman.sh(.bat)</td>
<td>Runs the TCMAN administrative utilities necessary to manage the Pacific Application Server. tcman.sh is a command-line wrapper for tools that are implemented in tcmanager.sh and it also manages the instance variables, $CATALINA_HOME and in $CATALINA_BASE.</td>
</tr>
<tr>
<td>tcmanager.sh(.bat)</td>
<td>Called by tcman.sh to implement the TCMAN administrative utilities necessary for managing the Pacific Application Server. This script is a wrapper for the built-in Tomcat utilities, including startup.sh, shutdown.sh, version.sh, and configtest.sh. In addition, TCMAN adds new administrative functionality specifically for the Pacific Application Server.</td>
</tr>
<tr>
<td>progress_product_setenv.sh(.bat)</td>
<td>Called by the Tomcat setenv script to set Progress product-specific environment variables and make them available to Java processes. If this file exists in $CATALINA_HOME/bin and in $CATALINA_BASE/bin, the file in $CATALINA_BASE takes precedence.</td>
</tr>
</tbody>
</table>

Development servers compared to production servers

Pacific 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 74
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 4: 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 28
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:

```
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.

```
<!-- 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 `parm-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.

```
<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.

```
<http pattern="/web/**"
    auto-config="true"
    use-expressions="true"
    create-session="stateless"
    disable-url-rewritings="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 Pacific 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:

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

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 5: 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 Pacific Application Server's `/conf` directory. Where noted, standard Tomcat configuration files were extended to support configuration of the Pacific 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 6: Pacific Application Server properties files

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>appserver.properties</td>
<td>An added Progress properties file that sets custom Pacific 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>
<tr>
<td>jvm.properties</td>
<td>A standard properties file that sets JVM options for the Java Virtual Machine that a PAS instance runs in.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> You cannot use tcman.sh(.bat) config to modify this file.</td>
</tr>
<tr>
<td>openedge.properties</td>
<td>An added Progress properties file that contains OpenEdge-specific settings for debugging, logging, and other properties. For more information, open the conf/openedge.properties.README file.</td>
</tr>
<tr>
<td>server.xml</td>
<td>A standard Tomcat configuration file that was tailored for the Pacific Application Server to use Java system properties that are defined externally configured in the catalina.properties and the appserver.properties files.</td>
</tr>
<tr>
<td>web.xml</td>
<td>Standard Tomcat configuration files that contain defaults common to all the Web applications deployed in the server.</td>
</tr>
<tr>
<td>context.xml</td>
<td></td>
</tr>
</tbody>
</table>

### Pacific Application Server properties

The following tables list the configuration properties for a Pacific 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 7: PAS properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>catalina.base</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>catalina.home</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>com.sun.management.jmxremote.access.file</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>com.sun.management.jmxremote.password.file</td>
<td>A readable/writable string containing the absolute path to where the PAS instance JMX console password is stored.</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>Property</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</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 “Host” 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 “ClusterManager” in the Apache Tomcat documentation.</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 “ClusterManager” 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 “Cluster” 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 “Channel/Membership” 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 “Channel/Membership” 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 “Channel/Membership” in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</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 “Channel/Membership” 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 “ClusterManager” 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 “Channel/Receiver” 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 “Channel/Receiver” in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.clust.recv.maxthreads</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 “Channel/Receiver” in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.clust.recv.nodelay</td>
<td>A readable/writable Boolean indicating whether the Tomcat cluster message receiver uses the TCP_NODELAY option. The possible values are true (option on) and false (option off). For more information, see “Channel/Receiver” in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.clust.recv.port</td>
<td>A readable/writable numeric value containing the Tomcat cluster multicast receiver port number. For more information, see “Channel/Receiver” in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.clust.recv.selectortimeout</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 “Channel/Receiver” in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>Property Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</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: /local/server/common/*.jar,/local/server/common/x.jar</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><code>psc.as.executor.minsparethreads</code></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><code>psc.as.http.connectiontimeout</code></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><code>psc.as.http.maxconnections</code></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><code>psc.as.http.maxqueue size</code></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>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-------------------------------------------------------------------------------------------------------</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 “Connector” 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 true (enabled) and false (disabled). For more information, see “Connector” 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 “Connector” 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 “Connector” in the Apache Tomcat documentation.</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 “Connector” 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 “Connector” 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 “Connector” 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 “Connector” 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.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>psc.as.os</td>
<td>A read-only string indicating the system PAS was installed on. The possible values are unix and windows.</td>
</tr>
<tr>
<td>psc.as.parent</td>
<td>A read-only string containing the path to the home server. If psc.as.alias is home, then psc.as.parent 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>
</tbody>
</table>

**Caution:** The value of this property is stored as cleartext.
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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. standard indicates a full PAS installation, and instance 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 psc.as.webappdir). The possible values are true (automatic unpacking) and false (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) CATALINA_BASE configuration value. For more information, see &quot;Resources&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.as.version</td>
<td>A read-only string containing the PAS release version in the format major-version.minor-version.patch-version.</td>
</tr>
<tr>
<td>psc.as.webappdir</td>
<td>A readable/writable string containing the path for where the PAS server finds and loads web applications and WAR files from. If the path is not absolute, it is relative to the CATALINA_BASE path for the server (instance). For more information, see &quot;Host&quot; in the Apache Tomcat documentation.</td>
</tr>
<tr>
<td>psc.dns.name</td>
<td>A read-only string containing the server's fully qualified DNS name. This property cannot be configured using TCMAN.</td>
</tr>
<tr>
<td>psc.host.name</td>
<td>A read-only string containing the server's configured host name. This property cannot be configured using TCMAN.</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 8: PAS features**

<table>
<thead>
<tr>
<th>Property</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>Property</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</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>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>
Pacific Application Server directories

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

Table 9: Pacific Application Server directory structure extensions

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$CATALINA_HOME/common/lib/</td>
<td>Contains general 3rd party libraries that are shared by a server, its instances, and its web applications.</td>
</tr>
<tr>
<td>$CATALINA_BASE/common/lib/</td>
<td>Contains general 3rd party libraries that are used by a single instance and its web applications.</td>
</tr>
<tr>
<td>$CATALINA_HOME/extras/</td>
<td>Contains the WAR files of the default Tomcat web applications, host-manager.war, manager.war, and ROOT.war. It can also contain the WAR files that support Progress products. (For example: oeabl.war and oemanager.war, which can optionally be deployed to an instance to support OpenEdge applications and management.)</td>
</tr>
</tbody>
</table>
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 Pacific Application Server that you installed.

Instances are independently running copies of the core Pacific 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 Pacific Application Server, rather than deploying to the Pacific 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 Pacific 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 Pacific 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 Pacific 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 Pacific 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 Pacific Application Server, rather than deploying to the Pacific 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 Pacific 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) Pacific 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 Pacific 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 Pacific 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 Pacific 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>Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>instances</td>
<td>Display all the instances created from the Pacific Application Server installed in $CATALINA_HOME.</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>
<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 /logs 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>

### Creating instances with TCMAN

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

- Install the core Pacific 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 Pacific 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>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--p port_num</td>
<td>Specify the TCP port that listens for HTTP messages. The default is 8080.</td>
</tr>
</tbody>
</table>
You can also activate these ports:

<table>
<thead>
<tr>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>–P port_num</td>
<td>Specify the TCP port that listens for HTTPS messages. The default is 8443.</td>
</tr>
</tbody>
</table>

See Create an instance (create) on page 140 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>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>host-manager.war</td>
<td>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.</td>
</tr>
<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 on page 123 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/acme1 and specifies its ports:

```
$: /psc/pashome/bin/tcman.sh create -p 8501 -P 8601 -s 8701 /psc/acme1
Server instance acme1 created at /psc/acme1
```

### See also

- Server actions on page 140
- Create an instance (create) on page 140
- Delete an instance (delete) on page 142
- Display and manage an instance's configuration (config) on page 143
- Display or modify the server features of an instance (feature) on page 145
- Clean up or archive server log files (clean) on page 147
- Display server instances (instances) on page 148
- Register an instance for tracking (register) on page 150
- Stop tracking an instance (unregister) on page 159
- Start an instance (start) on page 155
- Stop an instance (stop) on page 156
- Display server, OS, and runtime version information (version) on page 157
- Test a server configuration (test) on page 158

### 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 `SCATALINA_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 Pacific 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 Windows 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:

   ```sh
tcman service oepas1 register
   ```

   where `oepas1` 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:

   ```sh
tcman service oepas1 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 152
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 Pacific Application Server instance whether the server is running (online) or is not running (offline).</td>
</tr>
<tr>
<td>Action</td>
<td>Purpose</td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>undeploy</td>
<td>Remove a Web application from running (online) or stopped (offline) instances.</td>
</tr>
<tr>
<td>reload</td>
<td>Restart a deployed, running web application so that the application can pick up changes to its classes or libraries.</td>
</tr>
<tr>
<td>status</td>
<td>List information from the core server’s memory, including web application statistics.</td>
</tr>
<tr>
<td>leaks</td>
<td>List Web applications with potential memory leaks.</td>
</tr>
<tr>
<td>enable</td>
<td>Start a Web application that is deployed but not running.</td>
</tr>
<tr>
<td>disable</td>
<td>Stop a running Web application.</td>
</tr>
<tr>
<td>resources</td>
<td>List the global resources used by the core server.</td>
</tr>
<tr>
<td>sessions</td>
<td>Display how many sessions are active for the specified Web application, categorized by their duration.</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 Pacific 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**

```bash
$CATALINA_BASE/bin/deployREST.sh CustomerService.paar ROOT
```

**SOAP service deployment (deploySOAP.sh)**

**Purpose**

Deploy SOAP services.

**Syntax**

```bash
$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**

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

**Packaging web applications**

The Pacific 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.

---

For more information on Tomcat directory structure, see http://tomcat.apache.org/tomcat-7.0-doc/index.html.

---

**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
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 57.)
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:

```bash
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 55 and Tomcat load balancing on page 62 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, however, 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 Pacific Application Server for OpenEdge: Application and Migration Development Guide and OpenEdge Development: ABL Reference.

See also
Display Web application HTTP sessions (sessions) on page 139

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:

```
#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"
stickyssession=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 `stickyssession` property is not set. For example:

```
#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 62, where the running status of instances is monitored.

**See also**
- Requirements for Apache proxy load balancing on page 58
- Configuring the Apache server for proxy load balancing on page 59

**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 59
Configuring PAS for OpenEdge instances for proxy load balancing on page 60

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 60

**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 same PAS for OpenEdge instance (i.e. maintaining server-side context is not required), your entries in `httpd.conf` should resemble the following example:

```
#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:

```
#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"
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/) for documentation on your version of the Apache server.

**See also**
Configuring PAS for OpenEdge instances for proxy load balancing on page 60

**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:

```
[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 59 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>    -->
   ```
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 57, 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 63
Configuring PAS for OpenEdge for Tomcat load balancing on page 64
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.

3. **A workers.properties file.**
   
   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 TCMA N `workers` action.

   **See also**
   
   Create a Tomcat worker configuration file (workers) on page 160

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:

```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
```

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.

3. **Generating and deploying a worker.properties file.**

   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 140
Create a Tomcat worker configuration file (workers) on page 160

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 65

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 10: 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.
Security overview

The default security configuration for the core Pacific Application Server (PAS) includes some standard Tomcat functionality, along with Progress customizations and new features. For details, see the following topics:

- Pacific Application Server production server customizations
- HTTPS support
- The Spring Security Framework in PAS for OpenEdge
- JVM security manager
- Realms and roles
- Remote access filters
- Development server and production server security issues

Pacific Application Server production server customizations

Although it is based on Tomcat, the Pacific Application Server (PAS) is a web application server that is configured to function as a production server. In contrast, the standard Tomcat product download is configured as a development server.
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 that tailor PAS as a production server 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:** Although the default PAS is a production server, you can, some tailoring, run PAS as a development server. You can, for example, enable remote administration by deploying the manager applications to an instance, start an instance with a shutdown port enabled, and so on.

### HTTPS support

The Pacific Application Server (PAS) supports HTTP over a Secure Sockets Layer (HTTPS) by allowing 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/acmel
Server instance acmel created at /psc/acmel
```
In PAS (and in Tomcat as well) HTTPS port support is provided by the Secure Socket Layer (SSL) implementation of the Java Secure Socket Extension (JSSE).

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

Server access via HTTPS is dependent on having an SSL 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 a production server. Using the default certificate on a fully functional and deployed production server, is a serious security risk.

---

**The Spring Security Framework in PAS for OpenEdge**

The Pacific 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 Manger.

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 Pacific 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 Pacific 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>
<thead>
<tr>
<th>Table 11: PAS roles mapped to Tomcat roles</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PAS role</strong></td>
</tr>
<tr>
<td>ROLE_PSCAdmin</td>
</tr>
</tbody>
</table>
Realms and 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_PSCOper</td>
<td>admin-script</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 12: 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 Pacific 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:

```
<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.

Development server and production server security issues

The Pacific 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.
Web application security configurations

In Pacific 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 templates**

• **Applying a Spring Security model to a Web application**
Enabling Spring Security for APSV and SOAP Web applications

- Local configurations
- SPA security configurations
- LDAP security configurations
- SAML security configurations
- Container security configurations
- CORS support
- Single sign-on (SSO) support in PAS for OpenEdge

### Spring Security configuration templates

PAS for OpenEdge provides a number of Spring Security models for Web applications. The security models included with PAS for OpenEdge are implemented in the templates described in the table below.

**Note:** The Spring Security framework enables you to implement other types of authentication systems. For more information on implementing authentication systems, see the Spring Security framework documentation.

<table>
<thead>
<tr>
<th>Security Model</th>
<th>Configuration Template</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
<td>oeablSecurity-basic-local.xml</td>
<td>This is the default configuration model. The Basic configuration authenticates clients using HTTP Basic authentication for user accounts, based on the user information in the users.properties file of the Web application.</td>
</tr>
<tr>
<td></td>
<td>oeablSecurity-basic-ldap.xml</td>
<td>This configuration model authenticates clients using HTTP Basic authentication for user accounts, based on the user information stored in the directory governed by LDAP (Lightweight Directory Access Protocol).</td>
</tr>
<tr>
<td></td>
<td>oeablSecurity-basic-oerealm.xml</td>
<td>This configuration model authenticates clients using HTTP Basic authentication for user accounts managed by an OpenEdge application server.</td>
</tr>
<tr>
<td></td>
<td>oeablSecurity-basic-saml.xml</td>
<td>This configuration model authenticates clients using HTTP Basic authentication for external user accounts represented by a SAML token.</td>
</tr>
<tr>
<td>Security Model</td>
<td>Configuration Template</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Form</td>
<td>oeablSecurity-form-local.xml</td>
<td>This model authenticates clients using the generic HTTP Form login authentication for user accounts, based on the users.properties file of the Web application. For more information on managing local user accounts, see the Spring Security framework documentation.</td>
</tr>
<tr>
<td></td>
<td>oeablSecurity-form-ldap.xml</td>
<td>This model authenticates clients using the generic HTTP Form login authentication for user accounts stored in the directory governed by LDAP (Lightweight Directory Access Protocol).</td>
</tr>
<tr>
<td></td>
<td>oeablSecurity-form-oerealm.xml</td>
<td>This model authenticates clients using the generic HTTP Form login authentication for user accounts managed by an OpenEdge application server.</td>
</tr>
<tr>
<td></td>
<td>oeablSecurity-form-saml.xml</td>
<td>This configuration model authenticates clients using HTTP Form login authentication for external user accounts represented by a SAML token.</td>
</tr>
<tr>
<td>Anonymous</td>
<td>oeablSecurity-anonymous.xml</td>
<td>This is the default configuration model for Web applications. With the anonymous security model, you make the Web service public. That is, any user can make requests to the Web service.</td>
</tr>
<tr>
<td>Container</td>
<td>oeablSecurity-container.xml</td>
<td>This model integrates Spring Security framework with the authentication service of the Java container. The Java container authenticates, and the Spring Security framework controls the authorization to Web application resources. You apply security configurations in the server's realm implementation in conf/server.xml. For more information on container configuration, see the Apache Tomcat Realm Configuration documentation.</td>
</tr>
</tbody>
</table>

The templates that you can use to implement a security model are located in:

$CATALINA_BASE/webapps/webapp_name/WEB-INF
You choose the security model for the Web application in:
$CATALINA_BASE/webapps/webapp_name/WEB-INF/web.xml

Applying a Spring Security model to a Web application

You implement a Spring Security model by supplying a reference to its configuration template in a Web application's web.xml file.

To choose and apply a Spring Security configuration:
1. Open the Web application's $CATALINA_BASE/webapps/webapp_name/WEB-INF/web.xml file.
2. Find the <context-param> tag.
3. Uncomment one of the configuration file templates from the list of security configurations in the <param-value> code block.

For example, the following shows oeablSecurity-form-local.xml as the chosen security model template.

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>
    <!--
    /WEB-INF/oeablSecurity-basic-local.xml
    /WEB-INF/oeablSecurity-basic-ldap.xml
    /WEB-INF/oeablSecurity-basic-oerealm.xml
    /WEB-INF/oeablSecurity-form-ldap.xml
    /WEB-INF/oeablSecurity-form-oerealm.xml
    /WEB-INF/oeablSecurity-anonymous.xml
    /WEB-INF/oeablSecurity-container.xml
    /WEB-INF/oeablSecurity-form-saml.xml
    /WEB-INF/oeablSecurity-basic-saml.xml
    -->
    /WEB-INF/oeablSecurity-form-local.xml
  </param-value>
</context-param>
```

Note: You cannot uncomment more than one security model template.


Note:
- All security models, with the possible exception of oeablSecurity-anonymous.xml, require further configuration with regard to setting up user accounts, roles, authentication, and authorization.
- Initial configuration enables Spring Security for REST Web applications. Additional configuration is necessary if you want to enable Spring Security for the APSV and the SOAP transports.
- Additional configuration steps are required to set up CORS support and Single sign-on.

For information on additional Spring Security configuration tasks, follow the link to the appropriate topic listed below.
Enabling Spring Security for APSV and SOAP Web applications

After you apply and configure a Spring Security model for Web applications, Spring Security is only enabled for REST Web applications. Additional configuration is necessary if you want to enable Spring Security for APSV and SOAP Web applications.

By default, security is disabled for both APSV and SOAP transports. The following snippet from oeablSecurity-xxxx.xml shows that security is disabled as a result of the import of apsv-basic.xml and soap-none.xml:

```
<!-- To disable security for APSV, import apsv-basic.xml (and comment out import of apsv-basic.xml) -->
<!-- To enable security for APSV, import apsv-basic.xml (and comment out import of apsv-none.xml) -->
<b:import resource="apsv-basic.xml"/>
<!--<b:import resource="apsv-basic.xml"/> -->

<!-- To disable security for SOAP, import apsv-basic.xml (and comment out import of soap-basic.xml) -->
<!-- To enable security for SOAP, import apsv-basic.xml (and comment out import of soap-none.xml) -->
<b:import resource="soap-basic-local.xml"/>
<!--<b:import resource="soap-basic-local.xml"/> -->
```
The first step to enable security for APSV and/or SOAP Web applications is to comment out the defaults and import `apsv-basic.xml` and/or `soap-basic-local.xml`. In the following example, Spring Security is enabled for both APSV and SOAP:

```xml
<!-- To disable security for APSV, import apsv-none.xml (and comment out import of apsv-basic.xml)-->
<!-- To enable security for APSV, import apsv-basic.xml (and comment out import of apsv-none.xml) -->
<!-- <b:import resource="apsv-none.xml"/> -->
<b:import resource="apsv-basic.xml"/>

<!-- To disable security for SOAP, import apsv-none.xml (and comment out import of soap-basic.xml)-->
<!-- To enable security for SOAP, import apsv-basic.xml (and comment out import of soap-none.xml) -->
<!-- <b:import resource="soap-none.xml"/> -->
<b:import resource="soap-basic-local.xml"/>
```

After you enable security, you need to update the `apsv-basic.xml` and/or `soap-basic-local.xml` template files.

See also

- Applying a Spring Security model to a Web application on page 80
- Updating `apsv-basic.xml` on page 82
- Updating `soap-basic-local.xml` on page 83

### Updating `apsv-basic.xml`

After you have enabled Spring Security for the APSV transport in an `oeablSecurity-xxxx.xml` file, you can complete additional security configuration by updating the `apsv-basic.xml` file as follows:
1. Optionally update user access by adding to or modifying the following defaults:

```xml
<intercept-url pattern="/apsv/**" method="HEAD"
    access="hasAnyRole('ROLE_PSCUser')"/>
<intercept-url pattern="/apsv/**" method="GET"
    access="hasAnyRole('ROLE_PSCUser')"/>
<intercept-url pattern="/apsv/**" method="POST"
    access="hasAnyRole('ROLE_PSCUser')"/>
```

2. Optionally enable Single Sign-on by uncommenting the reference to OEPreauthfilter:

```xml
<!-- OpenEdge PRE AUTH Filter -->
<!-- USER EDIT: uncomment PRE_AUTH_FILTER filter to enable pre-auth -->
<custom-filter position="PRE_AUTH_FILTER"
    ref="OEPreauthfilter" />
```

3. If you enabled oeablSecurity-xxx-saml.xml, uncomment the references to samlProcessingFilter and OESamlSSOProcessingFilter:

```xml
<custom-filter before="BASIC_AUTH_FILTER"
    ref="samlProcessingFilter" />
<custom-filter before="FORM_LOGIN_FILTER"
    ref="OESamlSSOProcessingFilter" />```

See also
Enabling Spring Security for APSV and SOAP Web applications on page 81

### Updating soap-basic-local.xml

After you have enabled Spring Security for the SOAP transport in an oeablSecurity-xxxx.xml file, you can optionally update user access by adding to or modifying the following defaults in soap-basic.xml:

```xml
<intercept-url pattern="/soap/wsdl/**" method="GET"
    access="hasAnyRole('ROLE_PSCUser')" />
<intercept-url pattern="/soap/**" method="POST"
    access="hasAnyRole('ROLE_PSCUser')" />
```
Local configurations

After choosing a local security configuration model (oeablSecurity-XXXX-local.xml) for a Web application, you can add, remove, modify users or user authentication settings by updating the $CATALINA_BASE/webapps/webapp_name/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 hashed and salted passwords, and add the secure passwords to the users.properties file.

You can use the Open Source Jacksum utilities to generate hashed and salted passwords.

See also
Using Jacksum to generate hashed and salted passwords on page 85
Access control in local security configurations on page 86
Using Jacksum to generate hashed and salted passwords on page 85
Access control in local security configurations on page 86

Adding new users in local security configurations

To add a new user, update the users.properties with an entry in the following format:

```
username=password, ROLE1,[ ... , ROLEn, ] { enabled | disabled }
```

For example, the following shows the entries for the default users restuser and restdebug:

```
restuser=password,ROLE_PSCUser,enabled
restdebug=password,ROLE_PSCUser,ROLE_PSCDebug,enabled
```
Using Jacksum to generate hashed and salted passwords

To provide greater security for user accounts stored in users.properties, you can use the Open Source Jacksum utility to generate hashed and salted passwords for the entries in the file. For more information on hash and salt, search for hash and salt cryptography references on the Web.

To use the Jacksum utility:

1. Download and install the Jacksum software by following the instructions in http://www.jonelo.de/java/jacksum/#Installation.
2. Execute the following command to open the Jacksum installation directory:
   
   CD <installation-dir>\jacksum <version>

3. Execute the following command to get an encoded password for the required string. In this example, you are encoding the string newpassword{newuser}:
   
   java -jar jacksum.jar -a sha -E base64 -q "txt:newpassword{newuser}"

   In this example, you encoded the string newpassword{newuser}, and Jacksum returned the encoded password, Y9+iCjOORQoqoTWdRgjd/81Ms9w=

   **Note:**

   This example uses the sha algorithm to hash the newpassword string, and the newuser string in the curly braces as salt.

   You can use any of the popular hashing techniques that Jacksum supports, such as CRC, SHA, Tiger, sum and Whirlpool. And, you can use any string as salt in the encoding process.

5. Uncomment the code snippet that implements password hashing in the Web application.

   For example, assume that you are using the Basic security configuration file, oeablSecurity-basic-local.xml. Do the following:
a) Navigate to and uncomment the code snippet in bold in the
`oeablSecurity-basic-local.xml`:

```xml
<authentication-manager id="RestApplicationtAuth"
  <authentication-provider>
    <!-- Uncomment to add strong password hashing in
    users.properties
    <password-encoder hash="sha" base64="true">
      <salt-source user-property="username" />
    </password-encoder>
    <user-service properties="/WEB-INF/users.properties" />
  </authentication-provider>
</authentication-manager>
```

After uncommenting, the code snippet must look as follows:

```xml
<authentication-manager id="RestApplicationtAuth"
  <authentication-provider>
    <password-encoder hash="sha" base64="true">
      <salt-source user-property="username" />
    </password-encoder>
    <user-service properties="/WEB-INF/users.properties" />
  </authentication-provider>
</authentication-manager>
```

b) Save the security configuration file, `oeablSecurity-basic-local.xml`.

6. Add the new user account, `newuser`, and the encoded password (that you generated in Step 3 on page 85) to the `users.properties` file by appending the following line of code in it:

```
newuser=Y9+iCjOQRqoqTwDrRgjd/81Ms9w=,ROLE_PSCAdmin,enabled
```

In this example, you assigned the existing `ROLE_PSCAdmin` role to the `newuser` and enabled the user account `newuser` to work with the REST Management Agent.

**Note:** The `users.properties` file cannot store both plain text and encoded passwords. Therefore, before adding a new user with an encoded password, you must ensure that all the existing users and the users you add to the `users.properties` file have encoded passwords.

7. Save and close the `users.properties` file.

---

**Access control in local security configurations**

You can define access privileges to resources by adding Spring Security expressions in a `oeablSecurity-XXXX-local.xml` configuration file.
SPA security configurations

OpenEdge Single Point of Authentication (SPA) allows you to use an OpenEdge application server as a source of user account information for the authentication process. You implement SPA when you specify either oeablSecurity-basic-oerealm.xml or oeablSecurity-form-oerealm.xml as the Spring Security model in the Web application's web.xml file.

The OpenEdge SPA implementation extends the Spring Security authentication process. The implementation consists of 2 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:
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 14: 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>
This method validates the client-supplied password of the user account against the stored password.

**Note:** You can define both the clear-text and digest form of the `ValidatePassword()` method.

This method searches for the user account, validates whether the account domain is valid and enabled, and returns the account numeric ID.

**Note:** 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.

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 15: OERealmUserDetailsImpl module attributes**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Data type</th>
<th>Character encoding format</th>
</tr>
</thead>
</table>
| ATTR_ROLES        | string-extent | "" or "xxxx[,yyyy,zzzz]"
|                   |            | **Note:** 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 `oeabiSecurity-xxxxx-oerealm.xml` grant/deny access to URL/method combinations and will be used by the external security as part of its authorization functions. |

<p>| ATTR_ENABLED      | boolean   | A character from the <code>{T,t,F,f,Y,y,N,n,0,1}</code> set. |
|                   |           | <em>T,t,Y,y,1</em> denotes that the user is enabled. <em>F,f,N,n,0</em> denotes that the user is disabled. |</p>
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Data type</th>
<th>Character encoding format</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTR_LOCKED</td>
<td>boolean</td>
<td>A character from the {T,t,F,f,Y,y,N,n,0,1} set. T,t,Y,y,1 denotes that the user is locked. F,f,N,n,0 denotes that the user is not locked.</td>
</tr>
<tr>
<td>ATTR_EXPIRED</td>
<td>boolean</td>
<td>A character from the {T,t,F,f,Y,y,N,n,0,1} set. T,t,Y,y,1 denotes that the user is expired. F,f,N,n,0 denotes that the user is not expired. Spring Security does not allow expired users.</td>
</tr>
</tbody>
</table>

**Note:**
If your OERealm Service Interface uses attribute names different from the ATTR_XXXX attributes, update the attribute names in the OERealmUserDetails bean of the oeablSecurity-xxxx-oerealm.xml file.

Also note that some of the OE Realm Service Interface clients (like Rollbase, for example) use the default attribute names (i.e. in the ATTR_XXXX format) as shown in the table above. Do not change the attribute names if clients depend on the default names.

3. Build and test your OERealm service interface. You can perform this step using an ABL client test harness that calls the OERealm service interface.

**Note:** Errors returned to the client are generic. You can find error details that are more helpful in the application’s log file.

**See also**
SPA security considerations on page 95

### Updating the SPA configuration file

After choosing an SPA security configuration model, you must edit the SPA configuration file in order to update the properties of the authentication provider and the user details service. The authentication provider, OERealmAuthProvider, is a bean that defines how to create a ClientPrincipal after an OpenEdge database user account’s authentication process is successful. The user details service, OERealmUserDetails is a bean that specifies how the authenticated users connect to PAS for OpenEdge.

To update the SPA configuration file:
1. Open the SPA configuration file
   ($CATALINA_BASE/webapps/webapp_name/oeablSecurity-XXXX-oerealm.xml) and find the following element:

```
<b:bean id="OERealmAuthProvider"
    class="com.progress.rest.security.OERealmAuthProvider" />
```

2. Update the properties of OERealmAuthProvider.
   The following table describes the properties of OERealmAuthProvider:
### Table 16: OERealmAuthProvider properties

<table>
<thead>
<tr>
<th>Spring property</th>
<th>Default value</th>
<th>Range of accepted values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>createCOAuthn</td>
<td>&quot;true&quot;</td>
<td>&quot;true&quot;</td>
<td>&quot;false&quot;</td>
</tr>
<tr>
<td>key</td>
<td>&quot;&quot;</td>
<td>&quot;&lt;Valid string&gt;&quot;</td>
<td>Specifies the OpenEdge domain's access code to seal the ClientPrincipal token with the specified string.¹</td>
</tr>
<tr>
<td>userDomain</td>
<td>&quot;&quot;</td>
<td>&quot;&lt;Valid string&gt;&quot;</td>
<td>Specifies the OpenEdge domain name to append to the user account name if the name does not already contain a user domain.</td>
</tr>
<tr>
<td>multiTenant</td>
<td>&quot;false&quot;</td>
<td>&quot;true&quot;</td>
<td>&quot;false&quot;</td>
</tr>
<tr>
<td>authz</td>
<td>&quot;true&quot;</td>
<td>&quot;true&quot;</td>
<td>&quot;false&quot;</td>
</tr>
<tr>
<td>properties</td>
<td>&quot;empty&quot;</td>
<td>&quot;valid name and value&quot;</td>
<td>Loads static client-principal properties by name and value.</td>
</tr>
<tr>
<td>expires</td>
<td>&quot;0&quot;</td>
<td>&quot;0&quot; or &quot;positive integer&quot;</td>
<td>Expires ClientPrincipal in the specified number of seconds.</td>
</tr>
</tbody>
</table>

3. Find the following element:

```xml
<b:bean id="OERealmUserDetails" class="com.progress.rest.security.OERealmUserDetailsImpl"></b:bean>
```

4. Update the properties of OERealmUserDetails.

The following table describes the properties of OERealmAuthProvider:

---

¹ The `key` property can also be in the "oechl:<hex-string>" where `<hex-string>` is the output from the OE genpassword utility.
### Table 17: OERealmUserDetails properties

<table>
<thead>
<tr>
<th>Spring property</th>
<th>Default value</th>
<th>Range of accepted values</th>
<th>Description</th>
</tr>
</thead>
</table>
| realmURL        | value must be updated | • internal://localhost/nxgas — for a local PAS for OpenEdge instance  
                 • http[s]://host:port/oeabl_appname/apsv — for a remote PAS for OpenEdge instance  
                 • http[s]://host:port/aia_appname/aia — for a remote OpenEdge AppServer using AIA  
                 • AppServer[s]://Appserver_host:Appserver_port/service_name — for a remote OpenEdge Appserver configured with a NameServer  
                 • AppServerDC[s]://Appserver_host:Appserver_port/service_name — for a remote OpenEdge Appserver using Direct Connect | Specifies the URL of the OpenEdge application server where the realm is implemented.  
Note: The realm can be hosted on either a PAS for OpenEdge instance or an OpenEdge AppServer. This allows you to retain legacy implementations that were configured on an OpenEdge AppServer. |
| realmClass      | "OpenEdge. Security. Realm. HybridRealm" | "valid OOABL path" | Specifies the realm service interface's class path. SPA security configurations for the REST Web application must specify the `HybridRealm` interface class. |
| grantedAuthorities | "ROLE_PSCUser" | "ROLE_XXXX" | Specifies the static Spring roles granted for the user if the AppServer realm service interface does not support roles. |
| rolePrefix      | "ROLE_" | "valid string" | Specifies the role name's prefix that the Spring security framework must use when converting external role names into Spring roles names. |
| roleAttrName    | "ATTR_ROLES" | "valid string" | Specifies the string name supported by the realm service interface to return a comma-separated list of roles names. |
## Chapter 7: Web application security configurations

<table>
<thead>
<tr>
<th>Spring property</th>
<th>Default value</th>
<th>Range of accepted values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>enabledAttrName</td>
<td>&quot;ATTR_ENABLED&quot;</td>
<td>&quot;valid string&quot;</td>
<td>Specifies the string name supported by the realm service interface to return an account's enabled state.</td>
</tr>
<tr>
<td>lockedAttrName</td>
<td>&quot;ATTR_LOCKED&quot;</td>
<td>&quot;valid string&quot;</td>
<td>Specifies the string name supported by the realm service interface to return an account's locked state.</td>
</tr>
<tr>
<td>expiredAttrName</td>
<td>&quot;ATTR_EXPIRED&quot;</td>
<td>&quot;valid string&quot;</td>
<td>Specifies the string name supported by the realm service interface to return an account's expired state.</td>
</tr>
</tbody>
</table>
| realmPwdAlg         | "0"           | "0" | "3" | Specifies the format of the password passed to the realm service interface for validation. Specify:  
- 0 for clear-text passwords.  
- 3 for HTTP digest passwords. |
| realmTokenFile      | ""            | "" | "valid token file" | Optionally specify a file that holds a serialized ClientPrincipal used to authenticate the realm service interface. If left blank, authentication will not occur. |

5. Save and close the configuration file.
6. Test the OERealm service:
   a) In the WEB-INF/logging.xml file, uncomment the line following OEABL Security. For example:

   ```xml
   <!-- OEABL Security -->
   <logger name="com.progress.appserv.services.security" level="INFO"/>
   ```

   Note that you can change the level from INFO to DEBUG or TRACE to get more logging information.
   b) Start (or re-start) the PAS for OpenEdge instance.
   c) Check the contents of the $CATALINA_BASE/logs/web_appname_date.log file for startup errors related to your SPA security configuration.
   d) In a Web browser or any HTTP client, test the user authentication process.
SPA security considerations

The SPA security configuration process is used to authenticate users. The process itself 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 the SPA client, then the SPA 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 attributes.**

    You can enhance security by setting the following OERealmAuthProvider attributes while sealing the ClientPrincipal:

    - The `userDomain` attribute to set the static OpenEdge domain.
    - The `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 18: Available combinations of domain, domain access code, and user ID**

    | Domain name | Key attribute | User ID | Signing operation |
    |-------------|---------------|---------|-------------------|
    | "" (default) | "" (default) | "uid" | Seal the blank domain using the built-in Domain Registry blank access code.
    | "uid" | "uid@" | | |
    | "" | "" | | |

Pacific Application Server for OpenEdge: Administration Guide 95
<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;</td>
<td>&quot;&lt;xxxx&gt;&quot;</td>
<td>&quot;uid&quot; (&quot;uid@&quot;)</td>
<td>Seal the blank domain using the clear text sting value, &quot;&lt;xxxx&gt;&quot;.</td>
</tr>
<tr>
<td>&quot;&quot;</td>
<td>&quot;oech1::&lt;xxxx&gt;&quot;</td>
<td>&quot;uid&quot; (&quot;uid@&quot;)</td>
<td>Seal the blank domain using the encoded value, &lt;xxxx&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;xxxx&gt;&quot;</td>
<td>&quot;uid@abc&quot;</td>
<td>Seal the abc domain using the clear text sting value, &quot;&lt;xxxx&gt;&quot;.</td>
</tr>
<tr>
<td>&quot;abc&quot;</td>
<td>&quot;oech1::&lt;xxxx&gt;&quot;</td>
<td>&quot;uid@abc&quot;</td>
<td>Seal the abc domain using the encoded value, &lt;xxxx&gt;.</td>
</tr>
</tbody>
</table>

- **Set the multiTenant property.**

  If you enable the `multiTenant` attribute of the `OERealmAuthProvider`, all the user IDs are appended with the `userDomain` attribute value. In this case, the same key can used to seal the `ClientPrincipal` for all OE domains.

- **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 `ClientPrincipal`s 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` supercedes 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 its `enableCP` 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 realmTokenFile configuration property of OERealmUserDetails 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 "oech1::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 realmTokenFile configuration property of OERealmUserDetails 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.

**LDAP security configurations**

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.

If you choose one of the LDAP security models supported in PAS for OpenEdge (oeablSecurity-form-ldap.xml or oeablSecurity-basic-ldap.xml), you must update the settings shown in the following table:
Table 19: LDAP security configuration settings

<table>
<thead>
<tr>
<th>Find</th>
<th>Update</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;ldap-server id=&quot;PrimayLDAP&quot; url=&quot;ldap://enter_hostname:389/root_dn_here&quot; manager-dn=&quot;uid=admin,ou=system&quot; manager-password=&quot;admin&quot; /&gt;&gt;</td>
<td>The LDAP server's alias, URL, and the distinguished name (dn) and password of an LDAP account that has search and read access.</td>
</tr>
<tr>
<td>&lt;authentication-manager id=&quot;RestApplicationtAuth&quot; &gt;</td>
<td>All the properties of the LDAP authentication manager based on the design and implementation of your directory services</td>
</tr>
<tr>
<td>&lt;ldap-authentication-provider server-ref=&quot;PrimayLDAP&quot; group-role-attribute=&quot;cn&quot; role-prefix=&quot;ROLE_&quot; group-search-filter=&quot;(member={0})&quot; group-search-base=&quot;&quot; user-search-base=&quot;&quot; user-search-filter=&quot;(uid={0})&quot; /&gt;</td>
<td></td>
</tr>
</tbody>
</table>

**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.

**See also**
- CORS support on page 105

### SAML security configurations

Security Assertion Markup Language (SAML) is an XML-based standard for exchanging authentication and authorization data between business entities. It is a product of the OASIS Security Services Technical Committee. For more information on SAML, start with the SAML XML.org website.

If you choose one of the SAML security models supported in PAS for OpenEdge (oeablSecurity-form-saml.xml or oeablSecurity-basic-saml.xml), you must configure the security model after creating and registering Web application metadata. For more information, follow the link to the appropriate topic listed below.

**See also**
- Creating and configuring metadata files on page 99
- Registering the metadata file with an Identity Provider on page 100
- Updating the SAML configuration file on page 101
Creating and configuring metadata files

For each Web application (Service Provider), a metadata file must be generated and sent to all the Identity Providers that the user (Principal) wants to register with. A Service Provider metadata contains keys, services, and URLs that define its SAML endpoints.

To create and configure a Service Provider metadata file:

1. Extract the $DLC/servlets/rest/saml.jar file from the $DLC/servlets/rest/ directory, and then copy the contents to the WEB-INF folder of your REST Web application.
2. Create a new service provider metadata file for your Web application.

Note: You must generate your Service Provider metadata as expected by your Identity Provider.

The following is a sample Service Provider metadata file:

```xml
<!-- This is a demo Service Provider metadata file -->
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
  entityID="Entity ID">
  <md:SPSSODescriptor AuthnRequestsSigned="true" WantAssertionsSigned="true"
    protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
    <md:KeyDescriptor use="signing">
      <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:X509Data>
          <ds:X509Certificate>
            <Certificate information/>
          </ds:X509Certificate>
        </ds:X509Data>
      </ds:KeyInfo>
    </md:KeyDescriptor>
    <md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified</md:NameIDFormat>
    <md:AssertionConsumerService
      Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Location="<Location ID>" index="0" isDefault="true"/>
  </md:SPSSODescriptor>
</md:EntityDescriptor>
```
The Service Provider metadata file contains the following definitions (highlighted in bold in the previous codeblock):

<table>
<thead>
<tr>
<th>metadata</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>entityId</td>
<td>Specify the unique identity of the REST Web application (Service Provider).</td>
</tr>
<tr>
<td>AuthnRequestsSigned</td>
<td>Specify if the REST Web application (Service Provider) signs authentication requests.</td>
</tr>
<tr>
<td>WantAssertionsSigned</td>
<td>Specify if the REST Web application requires signed assertions.</td>
</tr>
<tr>
<td>Certificate</td>
<td>Specify the certificate that must be used by the IdP to register the Service Provider. This can either be a self-signed or a Certificate Authority (CA) signed certificate.</td>
</tr>
<tr>
<td>Binding</td>
<td>Specify the bindings to be included in the metadata for the WebSSO profile. Supported values are POST, Artifact, and PAOS. The order of bindings in the property determines the order of endpoints in the generated metadata. If the IdP does not require keys signed by a specific certification authority, then you can generate your own self-signed key using the Java utility “keytool”.</td>
</tr>
</tbody>
</table>

Note: The binding location does not have to be SSL-enabled.

3. Save and close the Service Provider metadata file.

Note: This metadata file is used to register the service provider with the IdP.

Registering the metadata file with an Identity Provider

After generating the [REST Web application (Service provider)] metadata file, you must register it with your identity provider (IdP). Each identity provider makes its metadata available for you to import into your service provider application.

An IdP administrator must provide you an IdP metadata file for your service provider based on the configurations that you specified in your service provider metadata file. Therefore, no changes are required in the IdP metadata file that you receive from your IdP administrator.

For instance, if you use the Progress Identity Provider, then you must register your REST Web application metadata file with the Progress Identity Provider and import the Progress identity provider file into your REST Web application as part of SAML specification. The Identity Provider metadata file is in the application WEB-INF folder.

To register your service provider metadata file with your Identity provider:

1. Send the Service Provider metadata file to your IdP administrator.
Note: The Service Provider metadata file is required to create an IdP metadata file.

2. Acquire the Identity Provider metadata file from your IdP administrator.
3. Copy the IdP metadata file in the WEB-INF folder of your REST Web application (Service Provider) in your Web server directory.

Updating the SAML configuration file

After registering the [REST Web application (Service provider)] metadata file with the Identity provider, you must configure your SAML security configuration to specify key management, [Service provider metadata], and Identity provider metadata details.

To configure your SAML security configuration model:

1. Open the SPA configuration file
   ($CATALINA_BASE/webapps/web_application/oeablSecurity-XXXX-saml.xml) and find the following code block:

   ```xml
   <b:bean id="keyManager" class="org.springframework.security.saml.key.JKSKeyManager">
     <b:constructor-arg value="WEB-INF/samlKeystore.keystore"/>
     <b:constructor-arg type="java.lang.String" value="password"/>
     <b:constructor-arg>
       <b:map>
         <b:entry key="<key-alias>" value="<password>"/>
       </b:map>
     </b:constructor-arg>
   </b:bean>
   ```

2. Update the following bean properties (highlighted in bold in the previous code block) as follows:

<table>
<thead>
<tr>
<th>Bean property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEB-INF/samlKeystore.keystore</td>
<td>Specify the keystore file that stores the keys for user authentication. This can be a self-signed or a Certificate Authority (CA) generated keystore.</td>
</tr>
<tr>
<td>key-alias</td>
<td>Specify the key alias, that is, a unique keystore name that you set during keystore generation.</td>
</tr>
<tr>
<td>password</td>
<td>Specify the keystore password required to authenticate the REST Web application.</td>
</tr>
</tbody>
</table>
Chapter 7: Web application security configurations

3. Search for the following code block to configure the metadata bean:
<b:bean id="metadata" class="org.springframework.security.saml.metadata.CachingMetadataManager">
    <b:constructor-arg>
        <b:list>
            <!-- IDP Metadata file -->
            <b:bean class="org.springframework.security.saml.metadata.ExtendedMetadataDelegate">
                <b:constructor-arg>
                    <b:bean class="org.opensaml.saml2.metadata.provider.FilesystemMetadataProvider">
                        <b:constructor-arg>
                            <b:value type="java.io.File">IdP metadata file name</b:value>
                        </b:constructor-arg>
                        <b:property name="parserPool" ref="parserPool"/>
                    </b:bean>
                </b:constructor-arg>
                <b:constructor-arg>
                    <b:bean class="org.springframework.security.saml.metadata.ExtendedMetadata">
                        <b:property name="local" value="false"/>
                        <b:property name="alias" value="default"/>
                        <b:property name="securityProfile" value="metaiop"/>
                        <b:property name="requireArtifactResolveSigned" value="false"/>
                        <b:property name="requireLogoutRequestSigned" value="false"/>
                        <b:property name="requireLogoutResponseSigned" value="false"/>
                        <b:property name="idpDiscoveryEnabled" value="false"/>
                        <b:property name="ecpEnabled" value="true"/>
                    </b:bean>
                </b:constructor-arg>
            </b:bean>
        </b:list>
    </b:constructor-arg>
    <b:bean class="org.springframework.security.saml.metadata.ExtendedMetadata">
        <b:property name="local" value="false"/>
        <b:property name="alias" value="default"/>
        <b:property name="securityProfile" value="metaiop"/>
        <b:property name="requireArtifactResolveSigned" value="false"/>
        <b:property name="requireLogoutRequestSigned" value="false"/>
        <b:property name="requireLogoutResponseSigned" value="false"/>
        <b:property name="idpDiscoveryEnabled" value="false"/>
    </b:bean>
</b:bean>

<!-- SP Metadata file -->
<b:bean class="org.springframework.security.saml.metadata.ExtendedMetadataDelegate">
    <b:constructor-arg>
        <b:bean class="org.opensaml.saml2.metadata.provider.FilesystemMetadataProvider">
            <b:constructor-arg>
                <b:value type="java.io.File">Service provider metadata file name</b:value>
            </b:constructor-arg>
            <b:property name="parserPool" ref="parserPool"/>
        </b:bean>
    </b:constructor-arg>
    <b:constructor-arg>
        <b:bean class="org.springframework.security.saml.metadata.ExtendedMetadata">
            <b:property name="local" value="true"/>
            <b:property name="alias" value="default"/>
            <b:property name="securityProfile" value="metaiop"/>
            <b:property name="requireArtifactResolveSigned" value="false"/>
            <b:property name="requireLogoutRequestSigned" value="false"/>
            <b:property name="requireLogoutResponseSigned" value="false"/>
            <b:property name="idpDiscoveryEnabled" value="false"/>
        </b:bean>
    </b:constructor-arg>
</b:bean>
4. Update the following bean properties (highlighted in bold in the previous code block) as follows:

<table>
<thead>
<tr>
<th>Bean property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDP metadata file name</td>
<td>Specify the location of the IdP metadata file.</td>
</tr>
<tr>
<td>Service provider metadata file name</td>
<td>Specify the location of the REST Web application (Service provider) metadata file.</td>
</tr>
</tbody>
</table>

5. Search for the following code block to configure the processing filter bean:

```xml
<bean id="samlProcessingFilter" class="com.progress.rest.security.OESamlProcessingFilter">
    <property name="samlEntryPointHandler" ref="initializeSAMLEntryPoint"/>
    <property name="authenticationManager" ref="RestApplicationtAuth"/>
    <property name="authenticationSuccessHandler" ref="successRedirectHandler"/>
    <property name="defaultFilterProcessesUrl" value="<Entity ID>"/>  
    <property name="defaultIdpProcessingUrl" value="<IdP URL>"/>  
</bean>
```

6. Update the following bean properties (highlighted in bold in the previous code block) as follows:

<table>
<thead>
<tr>
<th>Bean property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>defaultFilterProcessesUrl</td>
<td>Specify, as a value, the URL where the processing filter for WebSSO must be applied.</td>
</tr>
<tr>
<td>defaultIdpProcessingUrl</td>
<td>Specify, as a value, your IdP's URL. For instance, if you use the Progress IdP, you specify the value as <a href="https://secure-test.progress.com">https://secure-test.progress.com</a>.</td>
</tr>
</tbody>
</table>

7. Start (or re-start) your PAS for OpenEdge instance, in which the Web application is deployed, for the security configurations to take effect.
Container security configurations

If you chose the Container security (/WEB-INF/oeablSecurity-container.xml) as your security configuration model, you have enabled the Web application's use of the Tomcat container's authenticated user account and role definitions, which are defined in $CATALINA_BASE/conf/tomcat-users.xml.

Also, note that you must edit user accounts information in two configuration files, web.xml and oeablSecurity-container.xml, for authorization to Web resources.

To edit user accounts information in the web.xml file:

1. Search for the following comment: <!-- BEGIN:container.security-->

To edit user accounts information in the web.xml file, search for the BEGIN tag (<!-- BEGIN:container.security--> ) in the web.xml code and follow the instructions given by the NOTE, right below the BEGIN tag, in the code.

Note: The Spring Security configurations extends the Web application's security and uses the container's authenticated user accounts and assigned roles for authorization. So, if you chose the Container security model, you must edit user accounts information in two configuration files, web.xml and oeablSecurity-container.xml, for authorization to Web resources. To edit user accounts information in the web.xml file, search for <!-- BEGIN:container.security--> in the web.xml code and follow the . To edit user accounts information in the oeableSecurity-container.xml, see [XREF].

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 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 web.xml file. 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 security configuration (oeablSecurity-XXX.xml) that you applied to your Web application.
2. Find the OECORSFilter bean code snippet.

For example:

```xml
<b:bean id="OECORSFilter"
     class="com.progress.appserv.services.security.OECORSFilter"
>    <!-- Examples:
      <b:property name="allowAll" value="false"/>
      <b:property name="allowDomains" value="*"/>
      <b:property name="allowSubdomains" value="false"/>
      <b:property name="messageHeaders" value=""/>
      <b:property name="responseHeaders" value=""/>
      <b:property name="supportCredentials" value="true"/>
      <b:property name="maxAge" value="-1"/>
    -->
</b:bean>
```
3. Uncomment only the required properties in the OECORSFilter bean code snippet and update the value, if necessary.

Typically you only uncomment the allowAll and allowDomains properties, which allows default settings on other properties.

Usually, you set allowAll to false to ensure that all HTTP clients make CORS requests by sending an Origin header.

The allowDomains property limits CORS access to only certain HTTP clients.

Additionally, uncomment and use messageHeaders and responseHeaders properties if the HTTP clients require sending/receiving headers not contained in the defaults.

The following table is a list of all the OECORSFilter properties.

Table 20: OECORSFilter properties

<table>
<thead>
<tr>
<th>Property name</th>
<th>Description</th>
<th>Data types</th>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>allowAll&lt;sup&gt;2&lt;/sup&gt;</td>
<td>Specifies that CORS filter allow every client request. If this property is set to true, all the other CORS properties values are ignored by the CORS filter.</td>
<td>Boolean</td>
<td>true</td>
<td>true or false</td>
</tr>
<tr>
<td>allowDomains&lt;sup&gt;3&lt;/sup&gt;</td>
<td>Specifies the domains that can make server requests.</td>
<td>String</td>
<td>*</td>
<td>{ &quot;*&quot;</td>
</tr>
<tr>
<td>allowSubdomains</td>
<td>Specifies if subdomains of the permitted domains be allowed to make server requests.</td>
<td>Boolean</td>
<td>false</td>
<td>true or false</td>
</tr>
<tr>
<td>allowMethods&lt;sup&gt;4&lt;/sup&gt;</td>
<td>Specifies valid HTTP method names.</td>
<td>String</td>
<td>GET, PUT, POST, DELETE</td>
<td>Valid HTTP methods in upper case.</td>
</tr>
</tbody>
</table>

<sup>2</sup> To begin active CORS access control for all clients, turn allowAll property to false. If this property is set to true, the CORS filter grants access to all generic clients, that is, all generic requests (without CORS headers) are allowed to access the resources when the server is using non-standard HTTP or HTTPS ports.

<sup>3</sup> Ensure that you supply a fully qualified domain name. You must not append the path (/) separator in the domain name, and you must add a port (#) because the allowDomains default value is *, that is, all client requests are allowed to access the resources.

Ensure that you specify the list of domains in the allowDomains property for the filter to restrict all client requests from domains other than the domains listed in the allowDomains property.

<sup>4</sup> This list of method names is for all resources in the web application and needs to be coordinated with the resource authorization controls.
<table>
<thead>
<tr>
<th>Property name</th>
<th>Description</th>
<th>Data types</th>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>messageHeaders⁵</td>
<td>Specifies the message header to be passed as a header to the server. If you are passing multiple messages, you must specify a comma-separated list of messages.</td>
<td>String</td>
<td>Refer to the footnote.</td>
<td>Any valid string</td>
</tr>
<tr>
<td>responseHeaders⁶</td>
<td>Specifies the message header to be received by the client as a header from the server. If you are passing multiple messages, you must specify a comma-separated list of messages.</td>
<td>String</td>
<td>Refer to the footnote.</td>
<td>Any valid string</td>
</tr>
</tbody>
</table>

¹ This is for advanced HTTP client and server use. Use this only when the web application's clients are coded to use these headers. 

By default, the messageHeaders has the following methods: Accept, Accept-Language, Content-Language, Content-Type, X-CLIENT-CONTEXT-ID, Origin, Pragma, Cache-Control, Access-Control-Request-Headers, Access-Control-Request-Method.

⁶ This is for advanced HTTP client and server use. Use this only when the web application's clients are coded to use these headers. 

By default, the responseHeaders has the following methods: Cache-Control, Content-Language, Content-Type, Expires, X-CLIENT-CONTEXT-ID.
OECORSFilter properties

The OECORSFilter is a standard Spring security filter bean. It is declared in the Spring security configuration files (oeablSecurity-XXX.xml) in which its properties can be customized to override the defaults.

Progress Software recommends that you uncomment only the properties you need to set. The following table describes all the property names:

Table 21: OECORSFilter properties

<table>
<thead>
<tr>
<th>Property name</th>
<th>Description</th>
<th>Data types</th>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>allowAll</td>
<td>Specifies that CORS filter allow every client request. If this property is set to true, all the other CORS properties values are ignored by the CORS filter.</td>
<td>Boolean</td>
<td>true</td>
<td>true or false</td>
</tr>
</tbody>
</table>

4. Save the security configuration file.

Note: You must restart the Web server for the above security configuration updates to take effect.
### Property name | Description | Data types | Default | Range
--- | --- | --- | --- | ---
allowDomains | Specifies the domains that can make server requests. | String | * | {"*" | "domain1, domain2..."} |
allowSubdomains | Specifies if subdomains of the permitted domains be allowed to make server requests. | Boolean | false | true or false |
allowMethods | Specifies valid HTTP method names. | String | GET, PUT, POST, DELETE | Valid HTTP methods in upper case. |
messageHeaders | Specifies the message header to be passed as a header to the server. If you are passing multiple messages, you must specify a comma-separated list of messages. | String | Refer to the footnote. | Any valid string |
responseHeaders | Specifies the message header to be received by the client as a header from the server. If you are passing multiple messages, you must specify a comma-separated list of messages. | String | Refer to the footnote. | Any valid string |

---

9. Ensure that you supply a fully qualified domain name. You must not append the path (/) separator in the domain name, and you must add a port (#) because the allowDomains default value is *, that is, all client requests are allowed to access the resources.

Ensure that you specify the list of domains in the allowDomains property for the filter to restrict all client requests from domains other than the domains listed in the allowDomains property.

10. This list of method names is for all resources in the web application and needs to be coordinated with the resource authorization controls.

11. This is for advanced HTTP client and server use. Use this only when the web application’s clients are coded to use these headers. By default, the messageHeaders has the following methods: Accept, Accept-Language, Content-Language, Content-Type, X-CLIENT-CONTEXT-ID, Origin, Pragma, Cache-Control, Access-Control-Request-Headers, Access-Control-Request-Method.

12. This is for advanced HTTP client and server use. Use this only when the web application’s clients are coded to use these headers. By default, the responseHeaders has the following methods: Cache-Control, Content-Language, Content-Type, Expires, X-CLIENT-CONTEXT-ID.
### Property Names, Description, Data types, Default, Range

<table>
<thead>
<tr>
<th>Property name</th>
<th>Description</th>
<th>Data types</th>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>supportCredentials</td>
<td>Controls whether the CORS filter allows the client to send user credentials in the form of a COOKIE.</td>
<td>Boolean</td>
<td>true</td>
<td>true or false</td>
</tr>
<tr>
<td>maxAge</td>
<td>Specifies the maximum time (in seconds) for an application resource to be granted on request. After the specified time, the resource grant is revoked and the client must request access again. To set maximum time to infinity, set maxAge to -1.</td>
<td>Integer</td>
<td>-1</td>
<td>{-1</td>
</tr>
</tbody>
</table>

### 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.

This chapter describes how to configure the ClientPrincipal and how to enable SSO in the Pacific Application Server (PAS) for OpenEdge.

**Note:** In PAS for OpenEdge, SSO is available for client access via the APSV and REST transports but not for the SOAP transport.

### Configuring the ClientPrincipal

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. You can disable ClientPrincipal creation or modify the properties of the ClientPrincipal.

To configure the ClientPrincipal:

1. Identify and open the security configuration you applied to your Web application.
   
   Your Web application's security configuration is based on the security model specified for the Web application in $CATALINA_BASE/webapps/Web_application/WEB-INF/web.xml.

   **13** The default is true, which allows the client to use user logic sessions via COOKIES. If you do not want the client to send COOKIE user credentials, set the property value to false. By setting the property to false, you supply totally stateless resources to anonymous users.
In `web.xml`, find the `<context-param>` tag. Uncomment one of the `<param-value>` from the list of security configurations in the `<param-value>` code block.

2. In the configuration file for the relevant security model (`oeablSecurity-XXX.xml`), find the `OEClientPrincipalFilter` bean code snippet.

For example, the following shows code snippet in `oeablSecurity-basic-ldap.xml`:

```xml
<!-- The security filter that turns a Spring token into an OpenEdge ClientPrincipal object -->
<bean id="OEClientPrincipalFilter"
     class="com.progress.rest.security.OEClientPrincipalFilter">
    <!--
    <property name="enablecp" value="false" />
    <property name="domain" value="sample" />
    <property name="roles" value="sample" />
    <property name="authz" value="false" />
    <property name="expires" value="600" />
    <property name="acctinfo" value="true" />
    <property name="properties">
        <map>
            <entry key="prop-1" value="string1"/>
            <entry key="prop-2" value="string2"/>
        </map>
    </property>
    <property name="ccid" value="true" />
    <property name="anonymous" value="true" />
    <property name="appName" value="OE" />
    -->
</bean>
```

All the properties are commented out buy their default values are in effect. The following table describes the properties and their defaults:

**Table 22: OEClientPrincipal properties**

<table>
<thead>
<tr>
<th>Property name</th>
<th>Description</th>
<th>Datatypes</th>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>enablecp</td>
<td>Specifies if SSO is enabled or disabled.</td>
<td>Boolean</td>
<td>true</td>
<td>true or false</td>
</tr>
<tr>
<td>domain</td>
<td>Specifies Web application's domain name</td>
<td>String</td>
<td><code>&lt;webapp-name&gt;</code></td>
<td>ASCII</td>
</tr>
<tr>
<td>key</td>
<td>Specifies the access code for the domain</td>
<td>String</td>
<td>-</td>
<td>ASCII</td>
</tr>
<tr>
<td>roles</td>
<td>Specifies comma-separated list of ROLE names. You use this to perform Role-Based-Authentication (RBA) inside your business application.</td>
<td>String</td>
<td>-</td>
<td>ASCII</td>
</tr>
</tbody>
</table>

---

14 If you are using an anonymous security configuration model (`appServer-anonymous.xml`) for your Web application, you must ensure that the `roles` property value is left blank because if you do not mention any value for the `roles` property, the Web application assumes the default value, `ROLE_ANONYMOUS`. 
### Range Default Datatypes Description Property name

<table>
<thead>
<tr>
<th>Property name</th>
<th>Description</th>
<th>Datatypes</th>
<th>Default</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>authz</td>
<td>Specifies comma-separated list of ROLE names</td>
<td>String</td>
<td>-</td>
<td>ASCII</td>
</tr>
<tr>
<td>expires</td>
<td>Specifies the data expiration time in seconds.</td>
<td>String integer</td>
<td>0</td>
<td>0 to n</td>
</tr>
<tr>
<td>acctinfo</td>
<td>Specifies the Spring token account state properties</td>
<td>Boolean</td>
<td>-</td>
<td>true or false</td>
</tr>
<tr>
<td>properties</td>
<td>Specifies a map, a collection of property name entries.</td>
<td>Map</td>
<td>-</td>
<td>String property name and value</td>
</tr>
<tr>
<td>ccid</td>
<td>Specifies if CCID is enabled or disabled.</td>
<td>Boolean</td>
<td>true</td>
<td>true or false</td>
</tr>
<tr>
<td>anonymous</td>
<td>Specifies if the application server is being called from an anonymous Web application user or not.</td>
<td>Boolean</td>
<td>true</td>
<td>true or false</td>
</tr>
</tbody>
</table>

3. Uncomment any properties you want to change and update the `value` attribute.

Note that you can disable `ClientPrincipal` creation by uncommenting the following line:

```
<b:property name="enablecp" value="false"/>
```

4. Save the security configuration file.

**Note:** You must restart the Web server for security configuration updates to take effect.

---

### Enabling SSO for Web applications

To enable SSO, you must enable `OEPreauthfilter` Java bean in the configuration file of a Spring Security model that supports SSO and specify the domain where the `ClientPrincipal` will be validated.

---

15. The `properties` property is provided for you to pass any Spring token's state or any static information to PAS for OE. For instance, you can include Spring token's state, `isAcctLocked` and `isAuthenticated`, properties as entries in your properties property.

Note that, if you are using an anonymous security configuration model (`appServer-anonymous.xml`), Spring tokens do not get authenticated for any user account; which means that no Spring security state related entries in your properties property is passed to PAS for OpenEdge.

16. In the case of certain clients where `COOKIES` are not allowed and REST JSON data services do not perform URL rewriting of session-ids, the clients use CCID functionality (available to other application server clients) to obtain the user's login session-id. The CCID support inserts a `X-CLIENT-CONTEXT-ID` http header in each response message. The client may obtain the value and insert it into the next request's URL as a "JSESSIONID" query option.
The configuration templates that contain the OEPreauthfilter Java bean are:

- oeablSecurity-basic-ldap.xml
- oeablSecurity-basic-local.xml
- oeablSecurity-basic-oerealm.xml
- oeablSecurity-form-ldap.xml
- oeablSecurity-form-local.xml
- oeablSecurity-form-oerealm.xml

To enable SSO:

1. Open a oeablSecurity-xxx.xml configuration file and find the section labeled The Pre-auth security filter for SSO.

For example:

```xml
<!-- The Pre-auth security filter for SSO -->
<bean id="OEPreauthfilter" class="com.progress.appserv.services.security.OERequestHeaderAuthenticationFilter">

<!-- USER-EDIT: To turn on SSO:
1. Set "enabled" property to true
2. For single-domain supply the Domain Access Code as the value for "key" property
3. For multi-domain supply the absolute path of a 'registryFile' generated using
   OpenEdge's $DLC/bin/gendomreg.bat utility as "registryFile" property -->

   <property name="enabled" value="false"/>
   <property name="key" value=""/>
   <property name="registryFile" value="" />
   <property name="encoding" value="OECP"/>
   <property name="headername" value="X-OE-CLIENT-CONTEXT-ID"/>
   <property name="authenticationManager" ref="authenticationManager"/>
</bean>
```

2. Change the enabled property value from false to true.

```
<b:property name="enabled" value="true"/>
```

3. Set the domain(s) where the ClientPrincipal will be validated:

- For validating the token against a single domain, set the domain access code as the value attribute of the key property. The domain access code can be clear text or in the oecll::<hex-string> format.

  **Note:** You can generate the <hex-string> using the genpassword utility. For more information about the genpassword utility, see OpenEdge Getting Started: Installation and Configuration.

- For validating the token against multiple domains:
1. Create a domain registry file using the Generate Domain Registry `gendomreg.bat` utility that is located in the `$DLC\bin` folder.

2. Uncomment the `registryFile` property in the `OEPreauthfilter` bean code.

3. Set the `value` attribute of the `registryFile` property as the output file name generated by the `gendomreg.bat` utility.

**Note:** If you set values for both `key` and `registryFile` properties, the `ClientPrincipal` token is validated against multiple domains that are specified by the `registryFile` property value. To validate the token against a single domain, comment out the `registryFile` property in the `OEPreauthfilter` bean code.
Chapter 7: Web application security configurations
WebSpeed configuration and management

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

Note: Also see Modifying the WEB transport URL on page 28 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 Pacific 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 Pacific 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: Pacific 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 123
- **OEPROP** on page 21
- **REST API Reference for oemanager.war** on page 167
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 agentStartupParam parameter with the following syntax:

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

In addition, openedge.properties contains a ROOT.WEB section that 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:

\[
\text{instance}_\text{name}/\text{webapps/R}O\text{OT/\text{static}}
\]

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

\[
\text{instance}_\text{name}/\text{webapps/\text{webapp}_\text{name}/static}
\]

where \text{webapp}_\text{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:

\[
\text{instance}_\text{name}/\text{openedge}
\]
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=.
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 Pacific 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 Pacific Application Server.

See also

The tcman command on page 124
Extending TCMAN on page 126
Manager actions on page 129
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 Pacific 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 23: TCMAN general options

<table>
<thead>
<tr>
<th>Common options</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>-u user_name:password</td>
<td>Pass a valid user name and a password for HTTP Basic access authentication.</td>
</tr>
<tr>
<td>-v</td>
<td>Display verbose output.</td>
</tr>
<tr>
<td>-M URL</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>-B</td>
<td>Override default CATALINA_BASE environment settings.</td>
</tr>
<tr>
<td>-n</td>
<td>Debug the TCMAN action but do not execute changes.</td>
</tr>
<tr>
<td>-I instance_name</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 129
Server actions on page 140
Extending TCMAN on page 126
General actions on page 163

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 `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_osshell: 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_secmode: 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 -v command line option
  - tcman_debug: The state of the TCMAN -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 tcman command on page 124

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
/manager:stopped:0:manager
/oeadapters:stopped:0:oeabl
/oemanager:stopped:0:oemanager
/:stopped:0:ROOT
```

See also

Display Web application HTTP sessions (sessions) on page 139
Deploy a Web application (deploy) on page 131
Undeploy a Web application (undeploy) on page 132
Start a Web application (enable) on page 136
Stop a Web application (disable) on page 137
The tcman command on page 124

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.
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 157
Display detailed server status (status) on page 134
Test a server configuration (test) on page 158
The tcman command on page 124
Display or modify the server features of an instance (feature) on page 145
The tcman command on page 124

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

```
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 is online. It is not required if the server is offline.

-a app _name

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.

war_file_path

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 acmel instance of the core pashome server:

```
/psc/acmel/bin/tcman.sh deploy -a oeadapters /psc/pashome/extras/oeabl.war
OK - deployed /psc/pashome/extras/oeabl.war to local directory
/psc/acmel/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 129
Undeploy a Web application (undeploy) on page 132
Reload a Web application (reload) on page 133
Start a Web application (enable) on page 136
Stop a Web application (disable) on page 137
The tcman command on page 124

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.

-u user_id:password

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 131
List deployed applications (list) on page 129
Reload a Web application (reload) on page 133
The tcman command on page 124

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 136
Stop a Web application (disable) on page 137
List deployed applications (list) on page 129
The tcman command on page 124

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 $CATALINA_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.

-u user_name:password

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="48758784" usageMax="55967744" usageUsed="1525560"/>
    <memorypool name="PS Old Gen" type="Heap memory" usageInit="469762048" usageCommitted="1006632960" 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="3407872" usageMax="50331648" usageUsed="3303104"/>
    <memorypool name="PS Perm Gen" type="Non-heap memory" usageInit="67108864" usageCommitted="67108864" usageMax="67108864" usageUsed="67108864"/>
  </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="1" 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" query="/manager/status" queryString="XML=true" protocol="HTTP/1.1"/>
    </workers>
  </connector>
</status>
```
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**

```bash
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 129

The tcman command on page 124

---

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 137
The tcman command on page 124
The tcman command on page 124

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 136
The `tcman` command on page 124

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.
Manager actions

- 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:

```bash
$: /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 129

Server actions

This section details the actions available for creating and monitoring server instances.

See also

The tcman command on page 124

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] [-W pathname] [-N instance_name]
[-U user_id] [-G group_id] 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.

-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.

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 142
Display and manage an instance’s configuration (config) on page 143
Register an instance for tracking (register) on page 150
Start an instance (start) on page 155
Stop an instance (stop) on page 156

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.

-y

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/static
/PAS/wrkdir/acme3/webapps/ROOT/static/error
/PAS/wrkdir/acme3/webapps/ROOT/static/auth
/PAS/wrkdir/acme3/webapps/ROOT/META-INF
/PAS/wrkdir/acme3/webapps/ROOT/WEB-INF
/PAS/wrkdir/acme3/webapps/ROOT/WEB-INF/adapters
/PAS/wrkdir/acme3/webapps/ROOT/WEB-INF/adapters/rest/PingService
/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 140
Display server instances (instances) on page 148
The tcman command on page 124

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:

```bash
$: /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](https://docs.oracle.com/cd/E19879-01/821-0185/ablqz/index.html) for information about generating, exporting, and downloading a new server certificate):

```bash
$: /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:

```bash
$: /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=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 143
- Stop an instance (stop) on page 156
- The `tcman` command on page 124

## 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 `acmel`, an instance of the core server `pashome`, and save to a file:

```
/psc/pashome/tcman.sh clean -I acmel -A
```

See also
- The `tcman` command on page 124
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 150
Stop tracking an instance (unregister) on page 159
The tcman command on page 124

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, acm1 using the -v and -f options:

```
/psc/acm1/bin/tcman.sh plist -v
info: showing process ids for server 5942
5942 5963 5975 5988 6001 6015

/psc/acm1/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 154

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/acme1`:

```
/psh/pashome/bin/tcman.sh register test1 /psc/acme1
```

Notes

When you register an instance for tracking or create a new instance with the `create` command, an entry is created in the core Pacific 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.
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.

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><code>PR_DISPLAYNAME</code></td>
<td>Progress Pacific Application Server <code>alias_name</code></td>
</tr>
<tr>
<td><code>PR_DESCRIPTION</code></td>
<td>Progress Pacific 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=Pacific 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, acme1, then specify process ids to show detailed information.

```bash
/plsc/acme1/bin/tcman.bat plist -v
info: showing process ids for server with window title 13332
13332 14240

/plsc/acme1/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

/plsc/acme1/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
```
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 156

* The tcman command on page 124

## 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:

```
/pssc/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 155
- The tcman command on page 124

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

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 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 130
- The `tcman` command on page 124

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 acme1, which is an instance of the core server installed at /psc/pashome:

$$:
$:/psc/pashome/bin/tcman.sh -I acme1 test
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

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 124

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 150
The tcman command on page 124

Create a Tomcat worker configuration file (workers)

Purpose

Create a preliminary worker.properties file that supports the configuration of supporting servers (workers) for a Pacific 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 Pacific 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 25 on page 162.)

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 24: worker-list keywords

<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 worker.template plus entries for piw1 and piw2</td>
</tr>
<tr>
<td>piw1</td>
<td>Default entries from worker.template plus an entry for piw1</td>
</tr>
<tr>
<td>all</td>
<td>Default entries from worker.template plus entries for piw1 and piw2</td>
</tr>
</tbody>
</table>
The following is an example `workers.properties` file that includes entries for instances `piw1` and `piw2`:

**Table 25: worker.properties example**

```properties
# worker properties for instances rooted at U:\vobs\dlc\servers\pasw
# build date: 03/24/2014 11:09:23

# List of worker server instances
worker.list=piw1,piw2

#
# Global properties
#
# worker.maintain=60

#
# Common worker properties referenced by individual workers
#
worker.common.type=ajp13
worker.common.host=${psc.as.host.name}
worker.common.socket_timeout=10
worker.common.connect_timeout=10000
worker.common.socket_keepalive=true
worker.common.ping_mode=I
worker.common.ping_timeout=10000
worker.common.connect_timeout=0
worker.common.retry_interval=100
worker.common.recovery_options=7
# worker.common.connection_ping_interval=10000
# worker.common.fail_on_status=0# worker.common.max_packet_size=8192
# worker.common.recover_time=60

# properties for alias piw1 with jvmRoute piw1
worker.piw1.port=9996
worker.piw1.reference=worker.common

# properties for alias piw2 with jvmRoute piw2
worker.piw2.port=9996
worker.piw2.reference=worker.common
```

Note: When no registered instance is specified, all registered instances are automatically included.
Notes

- The tcman 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 124

Display help (help)

Purpose
Display summary or detailed help for all TCMAN actions, property names, and server features.

Syntax

```
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:

```
$: 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

```
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`:

```bash
/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 Pacific 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/](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 Pacific Application Server for OpenEdge sessions.

See also
Get session information on page 168
Get client information on page 169
Get current request information on page 171
Get configuration properties on page 172
Update configuration properties on page 173
Get runtime metrics on page 174
Reset runtime metrics on page 175
Cancel a request on page 176
Terminate a session on page 177

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
Get client information

Description
Query an ABL application to get a 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

```
{
    "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": "Iphz68lUQKexuHibyy6S+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",
  }
}
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":"oepas1",  "statusEnabled":"1",  "maxABLSessionsPerAgent":"200",  "agentLogFiles":"3",  "agentLoggingLevel":"3",  "ipver":"IPv4"  },  "outcome":"SUCCESS",  "errmsg":null,  "versionStr":"PASOE 11.5.0",  "versionNo":1}
"numInitialAgents":"1",
"agentLogFile":"/scratch/prmundra/17Sept/wrk/pasl/logs/oepasl.agent.log",
"collectMetrics":"1",
"agentExecFile":"/view/rdl115_linuxx86_64/vobs_prgs/linuxx86_64/dlc/bin/_mproapsv",

"idleResourceTimeout":"0",
"allowRuntimeUpdates":"0",
"connectionWaitTimeout":"3000",
"maxAgents":"10",
"publishDir":"/scratch/prmundra/17Sept/wrk",
"maxConnectionsPerAgent":"16",
"idleSessionTimeout":"300000",
"agentStartupParam":"-T /scratch/prmundra/17Sept/wrk/pasl/temp",
"statusEnabled":"1",
"maxABLSessionsPerAgent":"200",
"agentNumLogFiles":"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":"NA",
  "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":null,
    "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
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
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

```json
{
    "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 Pacific Application Server for OpenEdge agents.

See also

- List agents on page 178
- Get agent status information on page 179
- Get session metrics on page 180
- Get connection information on page 181
- Get request information on page 183
- Get configuration properties on page 187
- Update configuration properties on page 188
- Stop an agent on page 189

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
{
  "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
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-sCIVeFSQYmcb7RcHo10Zw/sessions

Request body example
NA

Response body example

NA

{ "result": { "threads": 6, "sessions": 5, "connections": 5, "requests": 16 }, "operation": "GET AGENT STATUS", "versionStr": "PASOE 11.5.0", "versionNo": 1, "outcome": "SUCCESS", "errmsg": "" }
Get connection information

**Description**
Get information on the connections of an agent of an ABL application.

**HTTP Operation**
GET
Chapter 10: REST API Reference for oemanager.war

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-sCIWesFSQYmcb7RcHo10Zw/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

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

```json
{
  "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
},
Agent management

```json
[{
  "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:getAgentSessionData",
  "SessionId": 3,
  "ConnectionId": 19,
  "StartTime": "2014-09-22T11:09:23.004",
  "EndTime": "2014-09-22T11:09:23.005",
  "RequestNum": 13
},
{
  "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",
```

Pacific Application Server for OpenEdge: Administration Guide 185
"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",
"RequestNum": 21
},
{
"RequestProcName":
"OpenEdge.Server.Admin.AgentInfo:getAgentRequestData",
"SessionId": 3,
"ConnectionId": 19,
"RequestNum": 22
},
{
"RequestProcName":
"OpenEdge.Server.Admin.AgentInfo:getAgentRequestData",
"SessionId": 3,
"ConnectionId": 19,
"StartTime": "2014-09-22T11:14:49.335",
"EndTime": "2014-09-22T11:14:49.336",
"RequestNum": 23
},
{
"RequestProcName":
"OpenEdge.Server.Admin.AgentInfo:getAgentRequestData",
"SessionId": 3,
"ConnectionId": 19,
"StartTime": "2014-09-22T11:14:49.690",
"EndTime": null,
"RequestNum": 24
}
],
"operation": "GET REQUESTS METRICS",
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
{
"result":{
"workDir": "/wrk/pas1/work",
"keyAliasPasswd": "",
"sslAlgorithms": "",
"agentMinPort": "62002",
"infoVersion": "9010",
"sessionDisconnProc": "",
"sslEnable": "0",
"applications": "pas1",
"PROPATH": "/wrk/pas1/openedge,/dlc/tty",
"collectStatsData": "0",
"agentStartupProcParam": "",
"agentShutdownProc": "",
"sessionShutdownProc": "",
"collectMetrics": "1",
"sessionStartupProcParam": "",
}
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
Stop an agent

Description
Stop an agent belonging to a 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"
}
```
Transport management

This section describes the REST services that are available for managing Pacific Application Server for OpenEdge SOAP, REST, and APSV transports.

See also
- Get transport configuration properties (SOAP/REST) on page 190
- Update properties of a transport (SOAP/REST) on page 192
- Enable or disable a transport (SOAP/REST) on page 193
- Enable or disable a transport (APSV) on page 194
- Get runtime metrics of a transport (SOAP/REST) on page 195
- Get runtime metrics of a transport (APSV) on page 197
- Reset runtime metrics for a transport (SOAP/REST) on page 198
- Reset runtime metrics of a transport (APSV) on page 199
- List deployed services (SOAP/REST) on page 200
- Get information about a service (SOAP/REST) on page 201
- Get configuration properties of a service (SOAP) on page 203
- Deploy a new service (REST) on page 204
- Deploy a new service (SOAP) on page 205
- Update runtime properties of a service (SOAP) on page 206
- Get runtime metrics for a service (SOAP) on page 207
- Reset runtime metrics for a service (SOAP) on page 208
- Enable or disable a service (SOAP) on page 209
- Remove a service (SOAP/REST) on page 210

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)**


**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",
"serviceFaultLevel": 2
},
"outcome": "SUCCESS",
"versionNo": 1,
"errmsg": "NA",
"versionStr": "PASOE 11.5.0",
"operation": "GET SOAP TRANSPORT PROPERTIES"
}

**Command-line example (REST)**


**Response body example (REST)**

{
"result": {
"appServerKeepalive": "denyClientASK,allowServerASK",
"idleSessionTimeout": 0,
"minSessions": 1,
"requestWaitTimeout": -1,
"initialSessions": 1,
"waitIfBusy": 0,
"staleO4GLObjectTimeout": 0,
"minIdleConnections": 0,
"connectionLifetime": 0,
"connectionTimeout": 0,
"transactionWaitTimeout": 0,
"maxTransactions": 0,
"initialTransactions": 1
}
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
  },
  "outcome":"SUCCESS",
  "errmsg":"NA",
  "versionStr":"PASOE 11.5.0",
  "versionNo":1
}

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
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


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)

curl -X GET -v

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"
  },
}
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,
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)

```json
{
  "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**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Success</td>
</tr>
<tr>
<td>500</td>
<td>Unexpected Server Error</td>
</tr>
</tbody>
</table>

**Command-line example**
```
curl -X DELETE -v 
http://localhost:16680/oemanager/applications/pas1/webapps/ROOT/transports/metrics
```

**Response body example**

```json
{
  "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:simpletestssf",
        "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",
  }
}
Command-line example (REST)


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": "1.1.5.0",
       "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:simpletestsfs",
        "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"
      }
    ]
  }
}
```
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,
}
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

```
-data-binary @PingService1.paar
-H "Accept: application/vnd.progress+json"
-H "Content-Type:application/vnd.progress.paar+zip"
-H "Content-Disposition: attachment; filename=PingService1.paar"
```

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
```
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
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**

```
```

**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
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/Web_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)**

```
```

**Response body example (SOAP)**

```
{
  "result": {
    "OESoapServiceDesc": {
      "status": "INITIALIZED",
      "descriptors": [],
      "oetype": "SOAP",
      "version": "11.4ALPHA",
      "description": "PAS OpenEdge SOAP Service.",
      "name": "SOAP",
      "state": "ENABLED",
```
"type": "OPENEDGE",
"uri": "http://localhost: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)
{"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"
   }
   ],
   "oetype":"REST",
   "version":"v11.4.0 (12-Dec-2013)",
   "description":"PAS OpenEdge REST Transport.",
   "name":"REST",
   "state":"DISABLED",
   "type":"OPENEDGE",
   "uri":"http://localhost.com:16680/\rest"
   }
   ],
   "operation": "UNDEPLOY REST TRANSPORT DESCRIPTOR",
   "versionStr": "PASOE 11.5.0",
   "versionNo": 1,
   "outcome": "SUCCESS",
   "errmsg": ""
}
Application management

This section describes the REST services that are available for managing ABL and Web applications deployed on the Pacific Application Server.

See also
List ABL application information on page 212
Get information about a specific ABL application on page 214
List Web applications deployed to an ABL application on page 216
Get information about a specific Web application on page 218

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": "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/"
}]}
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: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/_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/"
  }
],
"oetype": "WEBAPP",
"version": "11.5.0",
"description": "PAS OpenEdge WebApp",
"name": "ROOT",
"state": "STARTED",
"type": "OPENEDGE",
"uri": "http://localhost:16680/"}
List Web applications deployed to an ABL application

Description
Get information from the OEManager 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",
            ...
          }
        ]
      }
    ]
  }
}
```
"operation": "GET OEABL SERVICES"
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

```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:simpltestsf",
                "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 212, 214, 216, 218
ABL applications 52
Access control 86
Adding users 84
Administrative utilities 20, 24
Agent management 178
Agent status 179
Amazon load balancing 65
Apache load balancing 65
Apache proxy host load balancing 57
Apache proxy load balancing 58–59
Application management 212
apsv-basic.xml 82
Audience 11
Auto scaling 65

C
Cancel a request 176
clean action 147
Clients 169
Clusters 55
cfg action 143
Configuration files 30
Configuration properties 172–173, 187–188
Connection information 181
Container security configurations 105
CORS support 105
create 140
Creating instances 44

D
delete 142
deploy 131
Deployed services 200
deployREST.sh 50
deploySOAP.sh 51
Development server 74
Development servers 26
Directory structure extensions 40
disable 137

E
Elastic load balancing 65
enable 136
Enabling Spring Security 81
eve 164

F
feature action 145

H
help 163
HTTP sessions 53
HTTPS 70

I
info 130
instances 148
Instances
creating 44
managing 43
Running as a UNIX daemon 46
Running as a Windows service 47

J
Jacksum 85
JConsole 24
JMX 24
JVM security manager 71

L
LDAP security configurations 97
leaks 136
list 129
List agents 178
List clients 169
List sessions 168
Load balancing
Apache proxy host 57
Apache proxy load balancing 58
Tomcat load balancing 62
Logging 29

M
Managing instances 43
Managing Web applications 49
MBeans 24
MemoryRealm 72
REST API for oemanager.war (continued)
  get transport metrics 195, 179
  list agents 178
  list deployed services 200
  list Web applications 216
  removing a service 210
  reset runtime metrics 175
  reset service runtime metrics 208
  reset transport metrics 198–199
  sessions 168
  stop an agent 189
  terminate a session 177
  update configuration properties 173, 188
  update service runtime properties 206
  update transport properties 192

REST API Reference 167
REST service deployment (if needed)
  Nest index terms for sub entries 50
Roles
  assigning to users 72
ROOT Web application 27
Runtime metrics 174

S

SAML
  Creating and configuring service provider metadata 99
  Registering service provider 100
  Security configuration model 101
SAML security configurations 98
Security 69
service 152
Service deployment 204–205
Service enable or disable 209
Service information 201, 203
Service removal 210
Service runtime metrics 207–208
Service runtime properties 206
Session management 168
Session metrics 180
  sessions 139
Sessions 168
showproc 154
SOAP service deployment 51
soap-basic-local.xml 83
SPA configuration 90
SPA security configurations 87
Spring Security 71
SSO support 111
start 155
status 134
Sticky sessions 55
stop 156
Stop an agent 189
<table>
<thead>
<tr>
<th>T</th>
<th>Tcman 124</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCMAN</td>
<td>tcman 124</td>
</tr>
<tr>
<td>clean 147</td>
<td>TCMAN</td>
</tr>
<tr>
<td>config 143</td>
<td>clean 147</td>
</tr>
<tr>
<td>create 140</td>
<td>config 143</td>
</tr>
<tr>
<td>customizing 126</td>
<td>create 140</td>
</tr>
<tr>
<td>delete 142</td>
<td>customizing 126</td>
</tr>
<tr>
<td>deploy 131</td>
<td>delete 142</td>
</tr>
<tr>
<td>disable 137</td>
<td>deploy 131</td>
</tr>
<tr>
<td>enable 136</td>
<td>disable 137</td>
</tr>
<tr>
<td>env 164</td>
<td>enable 136</td>
</tr>
<tr>
<td>extending 126</td>
<td>env 164</td>
</tr>
<tr>
<td>feature 145</td>
<td>extending 126</td>
</tr>
<tr>
<td>general actions 163</td>
<td>feature 145</td>
</tr>
<tr>
<td>help 163</td>
<td>general actions 163</td>
</tr>
<tr>
<td>info 130</td>
<td>help 163</td>
</tr>
<tr>
<td>instances 148</td>
<td>info 130</td>
</tr>
<tr>
<td>leaks 136</td>
<td>instances 148</td>
</tr>
<tr>
<td>list 129, 149</td>
<td>leaks 136</td>
</tr>
<tr>
<td>manager actions 129</td>
<td>list 129, 149</td>
</tr>
<tr>
<td>register 150</td>
<td>manager actions 129</td>
</tr>
<tr>
<td>reload 133</td>
<td>register 150</td>
</tr>
<tr>
<td>resources 138</td>
<td>reload 133</td>
</tr>
<tr>
<td>server actions 140</td>
<td>resources 138</td>
</tr>
<tr>
<td>service 152</td>
<td>server actions 140</td>
</tr>
<tr>
<td>sessions 139</td>
<td>service 152</td>
</tr>
<tr>
<td>showproc 154</td>
<td>sessions 139</td>
</tr>
<tr>
<td>start 155</td>
<td>showproc 154</td>
</tr>
<tr>
<td>status 134</td>
<td>start 155</td>
</tr>
<tr>
<td>stop 156</td>
<td>status 134</td>
</tr>
<tr>
<td>test 158</td>
<td>stop 156</td>
</tr>
<tr>
<td>undeploy 132</td>
<td>test 158</td>
</tr>
<tr>
<td>unregister 159</td>
<td>undeploy 132</td>
</tr>
<tr>
<td>using 123</td>
<td>unregister 159</td>
</tr>
<tr>
<td>version 157</td>
<td>using 123</td>
</tr>
<tr>
<td>worker 160</td>
<td>version 157</td>
</tr>
</tbody>
</table>

| Terminate a session 177 |
| test 158 |
| Tomcat load balancing 62–64 |
| Transport management |
| URLs 27 |
| Transport metrics 195, 197–199 |
| Transport properties 190, 192 |
| Transports 193–194 |

<p>| U |
| undeploy 132 |
| UNIX daemon 46 |
| unregister 159 |</p>
<table>
<thead>
<tr>
<th>URLs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEB transport 28</td>
</tr>
</tbody>
</table>

| V |
| version 157 |

| W |
| Web application deployment 50 |
| Web application security configurations 77 |
| Web applications |
| managing 49 |
| Packaging 51 |
| WEB transport URL 28 |
| WebSpeed |
| configuration 117 |
| configuring openedge.properties 119 |
| management 117 |
| managing application files 120 |
| security 119 |
| Windows service 47 |
| worker 160 |