Progress® OpenEdge® Business Process Server: Customization Guide
## Table of Contents

**Preface**
- About this documentation ................................................................. 9
- User types .......................................................................................... 9
- Information on documentation .......................................................... 10
- Conventions used in this manual ....................................................... 11
- Product support contact information ............................................... 12

**Chapter 1: OpenEdge Business Process Server overview**
- Business Process Server components ............................................ 16
- How Business Process Server works ............................................. 18
- Business Process Server user types .............................................. 20

**Chapter 2: Customizing Properties files**
- Customizing Support Group’s contact details .................................. 22
- Customizing the date format in Business Process Portal .................. 23

**Chapter 3: Customizing password security**
- Implementing customized Password security ................................ 25
  - IPasswordSecurityData ................................................................. 26
  - PasswordSecurityService ......................................................... 28
  - Password security interfaces .................................................... 28
  - Password Security framework interfaces .................................. 29
  - Password encryption ................................................................. 33
  - Password encrypter framework ............................................... 33
  - Encrypting a password .............................................................. 34
    - Encrypting a password using API ........................................... 34
    - Encrypting a password using command .................................. 34
  - Enabling debug ........................................................................ 34

**Chapter 4: Customizing the Home module**
- The BPM CustomUI API ................................................................. 35
- JSPs ................................................................................................. 36
- Servlets ............................................................................................ 38

**Chapter 5: Customizing Login pages using BPM Sign-on API**
- Reviewing requirements for a single Login page ......................... 39
<table>
<thead>
<tr>
<th>Chapter 6: Customizing User Management Java interface</th>
<th>49</th>
</tr>
</thead>
<tbody>
<tr>
<td>The User Management Java interface..........................</td>
<td>49</td>
</tr>
<tr>
<td>Customizing the userManager interface.......................</td>
<td>50</td>
</tr>
<tr>
<td>LDAP realm customization.........................................</td>
<td>50</td>
</tr>
<tr>
<td>OEHybrid realm customization....................................</td>
<td>51</td>
</tr>
<tr>
<td>Manipulating customized User Management ....................</td>
<td>51</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 7: Customizing the Business Process Portal ........</th>
<th>53</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Process Portal features................................</td>
<td>54</td>
</tr>
<tr>
<td>BPM Single Sign-on API.............................................</td>
<td>54</td>
</tr>
<tr>
<td>User Manager and Access Control Management..................</td>
<td>55</td>
</tr>
<tr>
<td>User permissions....................................................</td>
<td>55</td>
</tr>
<tr>
<td>The Portal APIs......................................................</td>
<td>55</td>
</tr>
<tr>
<td>Portal customization................................................</td>
<td>55</td>
</tr>
<tr>
<td>The Login page.......................................................</td>
<td>56</td>
</tr>
<tr>
<td>Login Error page....................................................</td>
<td>59</td>
</tr>
<tr>
<td>Using the Shortcut keys............................................</td>
<td>59</td>
</tr>
<tr>
<td>Adding a new shortcut..............................................</td>
<td>60</td>
</tr>
<tr>
<td>Modifying an existing shortcut...................................</td>
<td>60</td>
</tr>
<tr>
<td>Hiding the Menu bar..................................................</td>
<td>60</td>
</tr>
<tr>
<td>Customizing a workstep to use concurrent processing.......</td>
<td>61</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 8: Customizing the Business Process Portal look and feel</th>
<th>63</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screen layout............................................................................</td>
<td>64</td>
</tr>
<tr>
<td>Modifying stylesheets..........................................................</td>
<td>65</td>
</tr>
<tr>
<td>Customization guidelines......................................................</td>
<td>66</td>
</tr>
<tr>
<td>Modifying the files..............................................................</td>
<td>67</td>
</tr>
<tr>
<td>Customized view......................................................................</td>
<td>69</td>
</tr>
</tbody>
</table>
Preface

For details, see the following topics:

- About this documentation
- User types
- Information on documentation
- Conventions used in this manual
- Product support contact information

About this documentation

This guide is part of the documentation set for Progress OpenEdge Business Process Server.

User types

Progress OpenEdge Business Process Server is a business process management system that can be used by the following types of users:

<table>
<thead>
<tr>
<th>User type</th>
<th>Responsibilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>User type</td>
<td>Responsibilities</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Manager</td>
<td>Responsible for automating business processes in a particular business domain. Business Process Portal's Management module serves as the primary interface to Business Process Server for the Manager, enabling the manager to monitor, analyze, and control business processes. Also uses the Business Process Modeler for modeling and simulation.</td>
</tr>
<tr>
<td>Application Developer</td>
<td>Responsible for creating customized applications for implementing business processes and developing interfaces associated with tasks. Application developers may work closely with Managers to define the requirements of an application, and determine the business processes.</td>
</tr>
</tbody>
</table>

**Information on documentation**

This documentation includes information for the entire range of Progress OpenEdge Business Process Server users. In the following table, we recommend the guides that are most relevant to each type of user.

<table>
<thead>
<tr>
<th>If you are the ...</th>
<th>Read the ...</th>
</tr>
</thead>
</table>
                     | *First Steps Guide*  
                     | *Terminology Guide* |
| Manager           | *Business Process Portal Manager's Guide*  
                     | *Business Process Portal User's Guide*  
                     | *Terminology Guide* |
### Conventions used in this manual

This document uses the following conventions and terminology notations.

<table>
<thead>
<tr>
<th>Convention (styles and terms)</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>bold</strong></td>
<td>Indicates titles of command buttons, check boxes, options, lists, dialog boxes, and portal page names.</td>
</tr>
<tr>
<td><strong>file path</strong></td>
<td>Indicates folder paths and file names.</td>
</tr>
</tbody>
</table>
Convention (styles and terms) | Purpose
--- | ---
*italic* | Indicates book titles.

*monospace* | Represents code segments or examples.

*backward slash "\" | Indicates the path in Windows environment. For UNIX environment, replace with forward slash "/"

OE_BPS_HOME or %OE_BPS_HOME% | Represents the installation folder of Business Process Server, C:\Progress\OpenEdge\oebpm\server.

STUDIO_HOME or %STUDIO_HOME% | Represents the installation folder of OpenEdge BPM components, C:\Progress\OpenEdge\oebpm\studio.

JB_OSS_HOME or %JB_OSS_HOME% | Represents the installation folder of JBOSS server, C:\Progress\OpenEdge\oebpm\jboss.

---

**Product support contact information**

If the product documentation does not provide a solution to your specific issue, or if you need clarification on the issue, then contact our Product Support team. You can contact the team through the Internet, telephone, or postal mail, as per the details provided in Table 1 on page 12.

**Table 1: Product Support Contact Information**

<table>
<thead>
<tr>
<th>To contact by</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web site</td>
<td><a href="http://progresslink.progress.com/supportlink">http://progresslink.progress.com/supportlink</a></td>
</tr>
<tr>
<td>If you are an existing customer, then you can log in to the above site for product support. If you are a first time user, then you need to create an account first.</td>
<td></td>
</tr>
<tr>
<td>Telephone¹</td>
<td>1-781-280-4999 for US, Latin America and Canada</td>
</tr>
<tr>
<td>1-781-280-4543 for the Product Support Fax Line</td>
<td></td>
</tr>
<tr>
<td>Postal Address¹</td>
<td>Progress Software Corporation</td>
</tr>
<tr>
<td>14 Oak Park Drive</td>
<td></td>
</tr>
<tr>
<td>Bedford, MA 01730, USA.</td>
<td></td>
</tr>
</tbody>
</table>

To enable us to quickly answer your questions, please provide the following information:

- Your name, installation site address and the license key for Business Process Server software.
- Your Business Process Server version and build number.

¹ For support telephone numbers and offices in your region, visit the support web site above. This contact information is for customer support only.
• Your operating system, application server and browser, with version and service pack details, if any.

• Your database management system and version, and information on JVM and JDBC used.
OpenEdge Business Process Server overview

Progress Software Corporation is a leading global provider of automated business process management solutions. The company’s product, Progress OpenEdge Business Process Server (henceforth referred to as Business Process Server or BP Server), is a comprehensive business process management platform, which enables companies to quickly transform their business processes into flexible and manageable Web applications, distributed over intranets, extranets, and the Internet.

Business Process Server addresses every stage in the business life cycle: define, integrate, publish, monitor, analyze, improve, and control. By adopting an end-to-end approach, Business Process Server incorporates all the key elements required to meet the ever-changing demands of e-business while ensuring e-business success. By providing integrated management tools, Business Process Server lets you monitor operations proactively, modifying automated processes dynamically based on changing external operations online. An overview of a typical automated business process management solutions is shown in Figure 1 on page 16.
Business Process Server components

Business Process Server is a suite of integrated components that enables you to easily build intranet, extranet, and Internet applications and manage your e-business. Business Process Server consists of the following components as in Figure 2 on page 17:

- Business Process Server components
- How Business Process Server works
- Business Process Server user types
Figure 2: Business Process Server components

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Process Portal Home</td>
<td>The Home module of Business Process Portal is the primary interface for application users, enabling them to interact with Business Process Server applications. Users can complete entries to various tasks and applications, and link to the support infrastructure required to complete these tasks.</td>
</tr>
<tr>
<td>Business Process Portal Management</td>
<td>The Management module enables managers to query, report on, and control processes and resources, visible only to the managers.</td>
</tr>
<tr>
<td>Business Process Portal Administration</td>
<td>The Administration module enables Business Process Server Administrators to modify configuration parameters, manage user or group access control, and install or uninstall Business Process Server applications.</td>
</tr>
<tr>
<td>Web services</td>
<td>This component allows application developers to: publish their applications as Web services, and find and convert other available Web services on the Internet into Business Process Server applications.</td>
</tr>
<tr>
<td>BPM Webflow</td>
<td>This component provides a framework for developing and implementing Web-enabled workflow applications.</td>
</tr>
</tbody>
</table>
# How Business Process Server works

The following figure provides an overview of the interaction between Business Process Server components.

**Figure 3: How Business Process Server works**
The following explanations correspond to the labels shown in the above figure, and describe how the components operate.

1. Progress Developer Studio for OpenEdge and Business Process Modeler provide an integrated development environment (IDE) for Business Process Server, where you can design and publish business processes. The application developer designs a process template (with the *.spt or *.swt extension) in the IDE that reflects the business flow and other business process requirements. Business rules for the process template can be defined using the Rule Editor, a BPM Events component that is launched with Progress Developer Studio for OpenEdge.


3. Once the process template is defined, Business Process Server Administrators use the Administration module to install the business process on the BP Server. Administrators can also configure Business Process Server components, manage user or group access control, and publish Business Process Server applications as Web services. Once installed, users access applications through servlets that pass the requests over an RMI/IIOP connection to the BP Server within an EJB Container.

4. The EJB Container provides a runtime environment that executes and manages Java-based program components that run on the server side of a client/server network. Within the EJB Container are the BP Server and BPM Events server.

5. The BP Server writes events to event tables in the database. Each Business Process uses JDBC to connect to database server as well as store events in the database. Within the BP Server, BPM Process Store uses JDBC to connect to the database server process and retrieve the events deposited by the BP Server process. BPM Process Store interprets the events and populates the process tables. These populated tables are used by Business Process Portal modules.

6. Once the process template is installed as a Business Process Server application, application users can use the Home module to do the following:
   - Access applications
   - Obtain information to perform their tasks
   - Launch the application to start process instances from the BP Server

7. Once the process template is installed as an Business Process Server application, managers can use the Management module (if they have access privileges) to monitor execution of process instances and create reports. Servlets receive requests from managers and pass them onto the BP Server over an RMI/IIOP connection. Managers use the Report Builder to define management reports that retrieve information through JDBC to the database server.

8. BPM Events is a rule-based event or message processing server that loads application rules and executes them against the BP Server and/or external events or messages. This server persists data in the database for recovery and with the help of JDBC connects to the database.

9. Managed Adapters exchange information between Business Process Server applications and external applications by converting Business Process Server-specific protocol to the protocol of an external system such as a database or ERP system. When users add a Managed Adapter to a work step, they can define complex mapping between Business Process Server dataslots and adapter inputs or outputs of the external application. At runtime when the work step is executed, the Managed Adapter sets the adapter inputs and configuration, and maps the outputs to the appropriate output dataslots.
10. BPM Webflow is a run-time component that executes the presentation flows. This component provides a Model, View, Controller (MVC) paradigm for developing presentation flow-based applications and executing them in a Web container.

11. Business Process Servers Web services component allows BP Server applications to be published as Web services.


Business Process Server user types

There are four user types within Business Process Server, Application users, Managers, Application developers, Business Process Server administrators.

User types

Each Business Process Server user type is defined below:

• **Application users** — Application Users use Business Process Server applications to coordinate specific business tasks with another department within their company, with another company within their organization, and/or with a business partner in another organization. The Home module in Business Process Portal serves as the primary interface in which Application Users run Business Process Server applications.

• **Managers** — Managers are typically experts in a particular business domain, such as quality assurance or human resources. They might need to work with managers from other groups in automating some of the business procedures that these groups share. The Management module in Business Process Portal serves as the primary Business Process Server interface for business managers to coordinate and integrate business processes, enabling them to exchange information with one another, and to share functionality over such standard communication channels as the Internet or e-mail.

• **Application developers** — Application developers are responsible for analyzing business processes and developing interfaces associated with creating tasks or processes. Application developers are often not domain experts themselves, but work closely with Managers to define business processes and determine the requirements of an application. Application developers use Progress Developer Studio for OpenEdge or Business Process Modeler to define the business process; the resulting process template file is tested, simulated, published, and run as a Business Process Server application.

• **Business Process Server administrators** — Business Process Server administrators are responsible for configuring Business Process Server components, managing user or group profiles and access control, and installing or uninstalling Business Process Server applications. The Administration module in Business Process Portal serves as the primary interface for Business Process Server Administrators to administer applications.

All Business Process Server user types can communicate by using one or more Business Process Server applications. They can also communicate with external applications.
Customizing Properties files

Properties files are available for all major Business Process Server components. You can use these files for customization of contact details or your date format; or for internationalization of labels, messages, and so on. For information on internationalization (or localization), see Localizing Business Process Server on page 107. A list of Properties files for each Business Process Server component is given in the following table.

Table 3: Properties files of each component

<table>
<thead>
<tr>
<th>Business Process Server component</th>
<th>Properties file(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OEBPS (Common for all components)</td>
<td>oebps_*_.properties</td>
</tr>
<tr>
<td></td>
<td>oebpsemail_*_.properties</td>
</tr>
<tr>
<td></td>
<td>oebpslog_*_.properties</td>
</tr>
<tr>
<td>BP Server</td>
<td>bpsserver_*_.properties</td>
</tr>
<tr>
<td></td>
<td>adapterframework_*_.properties</td>
</tr>
<tr>
<td></td>
<td>bpmprocessstore_*_.properties</td>
</tr>
<tr>
<td></td>
<td>dbadapter_*_.properties</td>
</tr>
<tr>
<td></td>
<td>emailadapter_*_.properties</td>
</tr>
<tr>
<td></td>
<td>fileadapter_*_.properties</td>
</tr>
<tr>
<td></td>
<td>ftpadapter_*_.properties</td>
</tr>
<tr>
<td></td>
<td>wsadapter_*_.properties</td>
</tr>
</tbody>
</table>
### Chapter 2: Customizing Properties files

<table>
<thead>
<tr>
<th>Business Process Server component</th>
<th>Properties file(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BPM Events</td>
<td>bpmevents_*.properties</td>
</tr>
<tr>
<td>BPM WebFlow</td>
<td>bpmwebflow_en.properties</td>
</tr>
</tbody>
</table>
| Progress Developer Studio for OpenEdge | Deployment_*.properties  
|                                   | designer_*.properties  
|                                   | documentgeneration_*.properties  
|                                   | FormEditor_*.properties  
|                                   | Rulewizard_*.properties |
| Business Process Portal           | BscResources.properties  
|                                   | bpmportal_*.properties  
|                                   | jasperreports_*.properties  
|                                   | UDDIBrowser_*.properties |

All these properties files are stored in the `OEBPS_HOME\conf\properties` directory. The format of all these files is as shown below:

*Key=Value*

For each key that is to be customized, you can specify its value. For example, if an organization wants to refer to a group of people working together on a task as a "Team", instead of "Group", then that organization can edit the `bpmportal.properties` file and change all "Group" references in values to "Team". For example, change `Create_Group=Create Group` to `Create_Group=Create Team`, change `GroupList=Group List` to `GroupList=Team List`.

**Note:** While modifying any properties file, change only the value.

For details, see the following topics:

- Customizing Support Group’s contact details
- Customizing the date format in Business Process Portal

### Customizing Support Group’s contact details

You can use the "mailURL" and "SupportEmail" keys from the `bpmportal.properties` file to add contact details of support team in your organization. In the `bpmportal.properties` file, you can change the value of the `mailURL` key from "mailto: support@progress.com" to the support team ID of your organization. Similarly, also change the value of the `SupportEmail` key from "support@mycompany.com" to the support team ID of your organization.
Customizing the date format in Business Process Portal

The Management module uses the Java class `SimpleDateFormat` to display times and dates, which you can configure using the `DateFormat` parameter in the `bpmportal.properties` file in your `%AppServer%\WEB-INF\classes\properties` directory. There may also be property files for each available language—for example, `bpmportal_en.properties` for English, `bpmportal_ja.properties` for Japanese in this directory. The `conf` parameter which specifies the `%AppServer%` location is param name="appserver.home". Under this system, use a time pattern string to specify the specific time and date format for your date format (in `bpmportal.properties`) or for a specific language (e.g., to change a date format only for Japanese, open the `bpmportal_ja.properties`). Additionally, you need to make the same changes in the `bpmportal.properties` and the property file for each available language in the `%OEBS_Home%\conf\properties` directory. ASCII letters are reserved as pattern specifiers in the `SimpleDateFormat` pattern, and the symbols are defined in the following table:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
<th>Presentation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>era designator</td>
<td>Text</td>
<td>AD</td>
</tr>
<tr>
<td>y</td>
<td>year</td>
<td>Number</td>
<td>1996</td>
</tr>
<tr>
<td>M</td>
<td>month in year</td>
<td>Text &amp; Number</td>
<td>July &amp; 07</td>
</tr>
<tr>
<td>d</td>
<td>day in month</td>
<td>Number</td>
<td>10</td>
</tr>
<tr>
<td>h</td>
<td>hour in am/pm (1~12)</td>
<td>Number</td>
<td>12</td>
</tr>
<tr>
<td>H</td>
<td>hour in day (0~23)</td>
<td>Number</td>
<td>19</td>
</tr>
<tr>
<td>m</td>
<td>minute in hour</td>
<td>Number</td>
<td>30</td>
</tr>
<tr>
<td>s</td>
<td>second in minute</td>
<td>Number</td>
<td>55</td>
</tr>
<tr>
<td>S</td>
<td>millisecond</td>
<td>Number</td>
<td>978</td>
</tr>
<tr>
<td>E</td>
<td>day in week</td>
<td>Text</td>
<td>Tuesday</td>
</tr>
<tr>
<td>D</td>
<td>day in year</td>
<td>Number</td>
<td>189</td>
</tr>
<tr>
<td>F</td>
<td>day of week in month</td>
<td>Number</td>
<td>2 (2nd Wed in July)</td>
</tr>
<tr>
<td>w</td>
<td>week in year</td>
<td>Number</td>
<td>27</td>
</tr>
<tr>
<td>W</td>
<td>week in month</td>
<td>Number</td>
<td>2</td>
</tr>
<tr>
<td>a</td>
<td>am/pm marker</td>
<td>Text</td>
<td>PM</td>
</tr>
</tbody>
</table>

Table 4: SimpleDateFormat time pattern strings
Refer to the following rules when using time and date formats:

- The number of pattern letters determines the format, according to the following:
  - (Text): If there are four or more pattern letters, use the full form. If there are less than four pattern letters, use the short or abbreviated form, if one exists.
  - (Number): The minimum number of digits. Shorter numbers are zero-padded to this amount. Year, however, is handled specially—if the count of ‘y’ is two, the Year will be truncated to two digits.
  - (Text & Number): Three or more, use text, otherwise use number.

- Characters in the pattern that are not in the ranges of [‘a’..'z'] and [‘A’..'Z’] are treated as quoted text. For instance, characters such as ‘:’, ‘,’ ‘‘’, ‘#’ and ‘@’ appear in the resulting time text even though they are not specified within single quotes.

- A pattern containing an invalid pattern letter results in an exception except during formatting.

The default date format setting for Business Process Server applications is MM/dd/yyyy, which displays a date of February 18, 2006 as 02/18/2006. The following table presents examples of the date format pattern using U.S. locales:

### Table 5: Format pattern examples for U.S. locales

<table>
<thead>
<tr>
<th>Format pattern</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;MM/dd/yyyy (default)</td>
<td>07/10/2006</td>
</tr>
<tr>
<td>&quot;yyyy.MM.dd G 'at' hh:mm:ss z&quot;</td>
<td>2006.07.10 AD at 15:08:56 PDT</td>
</tr>
<tr>
<td>&quot;EEE, MMM d ',yy&quot;</td>
<td>Mon, July 10, ’06</td>
</tr>
<tr>
<td>&quot;h:mm a&quot;</td>
<td>12:08 PM</td>
</tr>
<tr>
<td>&quot;hh 'o'&quot;clock' a, zzzz&quot;</td>
<td>12 o'clock PM, Pacific Daylight Time</td>
</tr>
<tr>
<td>&quot;K:mm a, z&quot;</td>
<td>0:00 PM, PST</td>
</tr>
<tr>
<td>&quot;yyyyy.MMMMMM.dd GGG hh:mm aaa&quot;</td>
<td>2006.July.10 AD 12:08 PM</td>
</tr>
</tbody>
</table>
Customizing password security

This chapter explains how to customize password security. For setting password security, see "Setting Password Security and Encryption" Chapter in Server Administrator’s Guide. For details, see the following topics:

- Implementing customized Password security
- IPasswordSecurityData
- PasswordSecurityService
- Password security interfaces
- Password Security framework interfaces
- Password encryption
- Password encrypter framework
- Encrypting a password
- Enabling debug

Implementing customized Password security

The default password policy implemented in Business Process Server provides limited functionality. You can customize the default password policy.

To customize the password security:
1. Implement the following three interfaces:
   - IPasswordSecurityData
   - IPasswordSecurityService
   - IPasswordRule

2. Place the custom implementation of the above-mentioned interfaces under one of the following locations.
   a) class files: $SBMWEBAPPDIR/WEB-INF/classes or
   b) jar file: $SBMWEBAPPDIR/WEB-INF/lib

3. Provide the names of the classes implementing the IPasswordSecurityData, and IPasswordSecurityService interfaces as values to the following parameters in bpmportal.conf respectively:

   **Table 6: Property**

<table>
<thead>
<tr>
<th>Property name</th>
<th>Property value</th>
</tr>
</thead>
<tbody>
<tr>
<td>bpmportal.ps.datahandler</td>
<td>Fully qualified name of class implementing IPasswordSecurityData interface.</td>
</tr>
<tr>
<td>bpmportal.ps.servicehandler</td>
<td>Fully qualified name of class implementing IPasswordSecurityService interface.</td>
</tr>
</tbody>
</table>

### IPasswordSecurityData

The following table provides you a brief description about the methods related to each policy provided in each of the interfaces.

**Policy and methods**

**Table 7: Policy and methods**

<table>
<thead>
<tr>
<th>Policy</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change At First Login</td>
<td>isChangeAtFirstLogin()</td>
<td>Checks if the policy to change the password at first login is enabled.</td>
</tr>
<tr>
<td>Password Expiration/Longevity</td>
<td>isLongevityEnabled()</td>
<td>Checks if the password longevity is enabled.</td>
</tr>
<tr>
<td></td>
<td>getLongevity(int type)</td>
<td>Provides the validity (in days) of the specified password type.</td>
</tr>
<tr>
<td>Password Hints(^2)</td>
<td>getPasswordHints()</td>
<td>Provides password hints (questions) which aids the user to recollect.</td>
</tr>
</tbody>
</table>

\(^2\) Currently these features are not supported and will be supported in future.
<table>
<thead>
<tr>
<th>Policy</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password Change Historya</td>
<td>isHistoryEnabled()</td>
<td>Checks if the policy to validate a new password against previously provided passwords in history, is enabled.</td>
</tr>
<tr>
<td></td>
<td>getHistoryCount()</td>
<td>Returns the number of previous passwords to be stored. Any new password created must be different from the old passwords which are stored.</td>
</tr>
<tr>
<td>Notification on Password Expiration</td>
<td>isNotifyEnabled()</td>
<td>Checks if the policy to notify the user about the password expiration.</td>
</tr>
<tr>
<td></td>
<td>getNotifyBeforeExpiration()</td>
<td>Returns the days before password expiration notification is to be send.</td>
</tr>
<tr>
<td>Account Lockout</td>
<td>isAccountLockoutEnabled()</td>
<td>Checks if the policy to lockout an account is enabled.</td>
</tr>
<tr>
<td></td>
<td>getMaxLoginAttempts()</td>
<td>Returns the maximum wrong login attempts allowed before the account is locked.</td>
</tr>
<tr>
<td></td>
<td>getLockoutDuration()</td>
<td>Returns the duration (in hours) after which a locked account will be unlocked.</td>
</tr>
</tbody>
</table>

It is recommended to use the following XML to manage the above data in their custom implementation so that it will be consistent with the metadata. See the XML provided in `IPasswordSecurityData` on page 26.

```xml
<Policies>
  <ChangeAtFirstLogin enable="true"/>
  <Longevity enable="true">
    <PasswordExpiration type="user" afterDays="90"/>
  </Longevity>
  <AccountLockout enable="true">
    <MaxLogInAttempts count="20"/>
    <LockoutDuration minutes="30"/>
  </AccountLockout>
  <Notify enable="true">
    <BeforeExpiration days="3"/>
  </Notify>
  <History count="5"/>
</Policies>

<PasswordHints>
  <Question>What is your mother's maiden name?</Question>
  <Question>What is your school name?</Question>
</PasswordHints>
```

**Note:** History, PasswordHints, and AccountLockout will be supported in future.

**Important:** The custom implementation should handle the reading and setting of the appropriate values in the implementation class.
PasswordSecurityService

The table below lists the parameters and their functions in password security service.

Method- Password Security Service

Table 8: Method- Password Security Service

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>isDueForChange(String userName, int userType, long lastModifiedTime)</td>
<td>Checks if the password of the specified user, which was last modified on the specified time, is due for change. The life of a password is determined by the type of the user.</td>
</tr>
<tr>
<td>isDueForNotify(String userName, long lastModifiedTime)</td>
<td>Checks if the time to notify the user about the password expiration is reached.</td>
</tr>
<tr>
<td>getPasswordSecurity()</td>
<td>Returns the password security data object stored within this service.</td>
</tr>
<tr>
<td>isChangeAtFirstLoginEnabled()</td>
<td>Checks if policy to change the password at first login is enabled.</td>
</tr>
<tr>
<td>canLockoutAccount(int incorrectCount)</td>
<td>Returns true if the maximum allowed wrong tries is less than or equal to the specified incorrect attempts count.</td>
</tr>
<tr>
<td>canUnlockAccount(long lockoutTime)</td>
<td>Returns true if the specified lockout time is greater than or equal to defined lockout duration. The specified value is the time elapsed after the account was locked.</td>
</tr>
<tr>
<td>validatePassword(String password)</td>
<td>Validates the specified password against the defined password policies and rules.</td>
</tr>
<tr>
<td>validatePassword(String password, List&lt;String&gt; history)</td>
<td>Validates the specified password against the defined password policies and rules. The password specified should be different from the passwords specified in the history.</td>
</tr>
<tr>
<td>getPasswordStrength(String password)</td>
<td>Returns the strength of the specified password. The password strength is the measurement of the effectiveness of a password as an authentication credential.</td>
</tr>
<tr>
<td>isWeakPassword(String password)</td>
<td>Returns true if the specified password does not meet the required safety and strength policies.</td>
</tr>
</tbody>
</table>

Password security interfaces

The following sections describe the password security interfaces.
**Note:** Class implementing 'IPasswordSecurityService' interface should have a constructor that takes 'IPasswordSecurityData' as an argument. For example,

```java
public CustomPasswordSecurityService(IPasswordSecurityData customSecurityData) {
    super(customSecurityData);
    ...
}
```

**IPasswordSecurityData**

Sample XML to capture the password policy metadata is as follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<PasswordSecurity>
    <Policies>
        <ChangeAtFirstLogin enable="true"/>
        <Longevity enable="true">
            <PwdExpiration type="user" afterDays="30"/>
        </Longevity>
        <AccountLockout enable="true">
            <MaxLogInAttempts count="20"/>
            <LockoutDuration minutes="30"/>
        </AccountLockout>
        <Notify enable="true">
            <BeforeExpiration days="3"/>
        </Notify>
        <History count="5"/>
        <PasswordHints>
            <Question>What is your mother's maiden name?</Question>
            <Question>What is your school name?</Question>
        </PasswordHints>
    </Policies>
    <Rules>
        <MinLength>10</MinLength>
        <MaxLength>64</MaxLength>
        <Charset atleast="3">
            <UpperCase count="1"/>
            <LowerCase count="1"/>
            <Numeric count="2"/>
            <SpecialChar count="1"/>
        </Charset>
        <!-- Starts with lower case alphabet-->
        <RegEx>^[A-Z].*$</RegEx>
    </Rules>
</PasswordSecurity>
```

**Note:** History, PasswordHints, and AccountLockout are not supported, however it will be supported in the future.

---

**Password Security framework interfaces**

The following sections describe the Interface Class Definitions for IPasswordSecurityData, IPasswordRule and IPasswordSecurityService.
package com.savvion.sbm.security.password;
import java.util.List;

/**<n* Defines the contract for password security data, providing methods
*n for getting details of password policy and rules.
*/

public interface IPasswordSecurityData {
    public static final int SYSTEM_PASSWORD = 1;
    public static final int USER_PASSWORD = 2;
    public static final int GUEST_PASSWORD = 3;
    /**
     * Checks if policy to change the password at first login is enabled.
     * @return true if this policy is enabled; false otherwise.
     */
    public boolean isChangeAtFirstLogin();
    /**
     * Checks if password longevity is enabled.
     * @return true if password expires after no. of days returned by
     * getLongevity(type)
     * @return false if password never expires
     */
    public boolean isLongevityEnabled();
    /**
     * Returns the life (in days) of the specified password type.
     * The valid password types are:<br>
     * <ul>
     * <li> {@link IPasswordSecurityData#SYSTEM_PASSWORD}
     * <li> {@link IPasswordSecurityData#USER_PASSWORD}
     * <li> {@link IPasswordSecurityData#GUEST_PASSWORD}
     * </ul>
     * @param type - One of the valid password type.
     * @return Life (in days) of the specified password type
     */
    public int getLongevity(int type);
    /**
     * Checks if policy to lockout an account is enabled.
     * @return true if this policy is enabled; false otherwise.
     */
    public boolean isAccountLockoutEnabled();
    /**
     * Returns the maximum wrong login attempts allowed before the account
     * is locked.
     * @return Maximum wrong login attempts allowed.
     */
    public int getMaxLoginAttempts();
    /**
     * The duration (in hours) after which a locked account will be unlocked.
     * @return Duration (in hours) after which a locked account will be
     * unlocked.
     */
    public int getLockoutDuration();
    /**
     * Checks if policy to notify the user about the password expiration
     * is enabled.
     * @return true if this policy is enabled; false otherwise.
     */
    public boolean isNotifyEnabled();
    /**
     * Returns the days before password expiration notification is to be send.
     * @return Days before password expiration notification is to be send.
     */
    public int getNotifyBeforeExpiration();
}
/**
 * Checks if policy to validate a new password against previously provided
 * passwords in history, is enabled.
 * @return true if this policy is enabled; false otherwise.
 */
public boolean isHistoryEnabled();
/**
 * Returns the number of previous passwords to be stored. Any new
 * password created must be different from the stored old passwords.
 * @return The number of previous passwords to be stored.
 */
public int getHistoryCount();
/**
 * Returns password hints (questions) which aids the user to recollect
 * a forgot password.
 * @return Hints to aid the user to recollect a forgot password.
 */
public List<String> getPasswordHints();
/**
 * Returns password rules object containing all the password rules.
 * This is used to validate a new password.
 * @return Password rule object (null if no rule is defined)
 */
public IPasswordRule getPasswordRule();

IPasswordSecurityService

package com.savvion.sbm.security.password;
import java.util.List;
/**
 * Defines the contract for different services provide to verify and
 * validate the password is compliant with the policy and rules.
 */
public interface IPasswordSecurityService {
/**
 * Returns the password security data object stored within this service.
 * @return IPasswordSecurityData - Password security data object.
 */
public IPasswordSecurityData getPasswordSecurity();
/**
 * Checks if policy to change the password at first login is enabled.
 * @return true if this policy is enabled; false otherwise.
 */
public boolean isChangeAtFirstLoginEnabled();
/**
 * Checks if the password of the specified user, which was last modified
 * on the specified time is due for change. The life of a password is
 * determined by the type of the user.
 * @return true if password is due for change; false otherwise
 */
public boolean isDueForChange(String userName, int userType,
                            long lastModifiedTime);
 Checks if the time to notify the user about the password expiration is reached. It is calculated using:
* The life of the Password i.e. Days after password expiration happens.
* Last modified time of password.
* Days before expiration, notification is to be send.

@return true if password is due for notify; false otherwise
/**
* Returns true if the maximum allowed wrong tries is less than or equal to the specified incorrect attempts count.
* @return true if the account can be locked; false otherwise
*/
public boolean isDueForNotify(String userName, long lastModifiedTime);
/**
* Returns true if the specified lockout time is greater than or equal to defined lockout duration. The specified value is the time elapsed after the account was locked
* @return true if the account can be unlocked; false otherwise
*/
public boolean canLockoutAccount(int incorrectCount);
/**
* Returns true if the specified password does not meet the required safety and strength policies.
* @param password - Password (Such that it is not null and not empty).
* @return - true if the password is weak; false otherwise
*/
public boolean validatePassword(String password);
/**
* Validates the specified password against the defined password policies and rules. The password specified should be different from the passwords specified in the history.
* @param password - Password (Such that it is not null and not empty).
* @param history - Password history (Ignored if null or empty)
* @throws RuntimeException - If the specified password is null or empty
*/
public void validatePassword(String password, List<String> history);
/**
* Validates the specified password against the defined password policies
* and rules. The password specified should be different from the passwords specified in the history.
* @param password - Password (Such that it is not null and not empty).
* @param history - Password history (Ignored if null or empty)
* @throws RuntimeException - If the specified password is null or empty
*/
public void validatePassword(String password, List<String> history);
/**
* Validates the specified password against the defined password policies
* and rules. The password specified should be different from the passwords specified in the history.
* @param password - Password (Such that it is not null and not empty).
* @param history - Password history (Ignored if null or empty)
* @throws RuntimeException - If the specified password is null or empty
*/
public void validatePassword(String password, List<String> history);
/**
* Validates the specified password against the defined password policies
* and rules. The password specified should be different from the passwords specified in the history.
* @param password - Password (Such that it is not null and not empty).
* @param history - Password history (Ignored if null or empty)
* @throws RuntimeException - If the specified password is null or empty
*/
public void validatePassword(String password, List<String> history);
/**
* Validates the specified password against the defined password policies
* and rules. The password specified should be different from the passwords specified in the history.
* @param password - Password (Such that it is not null and not empty).
* @param history - Password history (Ignored if null or empty)
* @throws RuntimeException - If the specified password is null or empty
*/
public void validatePassword(String password, List<String> history);
/**
* Validates the specified password against the defined password policies
* and rules. The password specified should be different from the passwords specified in the history.
* @param password - Password (Such that it is not null and not empty).
* @param history - Password history (Ignored if null or empty)
* @throws RuntimeException - If the specified password is null or empty
*/
public void validatePassword(String password, List<String> history);
/**
* Validates the specified password against the defined password policies
* and rules. The password specified should be different from the passwords specified in the history.
* @param password - Password (Such that it is not null and not empty).
* @param history - Password history (Ignored if null or empty)
* @throws RuntimeException - If the specified password is null or empty
*/
public void validatePassword(String password, List<String> history);
password encryption

It is important to store user passwords in a safe and secured manner in a computer system. One of the ways to have safe and secure password is to encrypt passwords before storing them. Encryption conceals information through code or cipher. There are many methods of encrypting data, each has its own set of characteristics. One of the commonly used encryption algorithm is DES (Data Encryption Standard), which is a two-way encryption algorithm (you can code and then decode a message, given the right keys), to encrypt your passwords.

In Business Process Server, the passwords stored in the files or in the database are in encrypted format.

password encrypter framework

Business Process Server provides a default out-of-the-box password encrypter. The default encrypter provided with Business Process Server uses “DES/ECB/PKCS5Padding” algorithm to encrypt and decrypt password. The key generation algorithm used is DES (Data Encryption Standard).
Encrypting a password

The encryption has to be performed on the factory class (PService), which is the entry point for this functionality. The following sub-sections help you to encrypt a password from a client code.

Encrypting a password using API

A password can be encrypted using the following syntax:

```java
// Encrypt the passed password
String encrPswd = Pservice.self().encrypt(pswd);
```

If the passed string is 'null' or 'empty string', then no operation is done and the passed string is returned. If there is any error during encryption then the generated error is thrown to the user.

Encrypting a password using command

A password can be encrypted using the `encrypt.cmd|sh` script present at `OEBPS_HOME/bin` folder. The usage is:

```
encrypt.cmd [-help] [string to encrypt]
where:
  -help : print this information
  string : String to be encrypted
eg:
  encrypt.cmd ebms
```

Enabling debug

Enabling the debug will provide more information about the encryption. This can be enabled by adding the property `oebps.password.debug=true` in `oebps.conf` available at `OEBPS_HOME/conf` folder.
Customizing the Home module

This chapter presents the methods to customize the Home module using BPM CustomUI API, templates and HTML frames.

For details, see the following topics:

- The BPM CustomUI API
- JSPs
- Servlets

The BPM CustomUI API

Business Process Server provides a public API for the Home module called the BPM CustomUI API, exposing some internal features to provide more ease and flexibility for you. The BPM CustomUI API provides a convenient mechanism for you to incorporate features of the Business Process Portal within customized applications and to create your own specialized portals.

BPM CustomUI API features

In addition, BPM CustomUI API development allows streamlining of the code used to generate the various displayed pages, contributing to the page clarity and enabling you to easily grasp what is necessary to make the modifications you need. Part of BPM CustomUI APIs migrate existing precompiled JSP servlets to full-fledged dynamic JSP pages making use of this API, to make the task of Home module customization easier.

Two primary benefits are derived from a BPM CustomUI API:
The BPM CustomUI API exposes all the necessary elements for you to create customized pages.

The current code that generates display pages may also use the BPM CustomUI API with the benefits of simplicity and clearer structure, allowing you to quickly understand the structure and make modifications to display pages. The design also exposes the HTML for each type of element, allowing for fine-grained customization of each building block display of the Business Process Portal pages.

The API methods can be used to access and interact with the data available to the Business Process Portal, as well as to retrieve presentation for it. The retrieved presentation is made even more flexible through the use of templates, which is explained at the end of this chapter.

The API is provided as a set of related classes. The main class, BizSiteBean, is the point of entry for all uses of the API. The other classes like, BizSiteApp, BizSiteTask, BizSiteDataslot, and BizSiteProcess, are returned by some of the API methods that are part of BizSiteBean and are respectively used to represent applications, tasks, dataslots, and process instances. These supplemental classes contain methods to manipulate the types of data that they represent.

All applications that intend to incorporate API features must access them through the BizSiteBean, meaning the main preparation that must be made in all such applications is to provide access to this class. Through this single point of entry, the API may be used from within a number of different types of applications. These include JSPs and servlets. Its use in these various contexts are explained.

### JSPs

If you want to use the API within a JSP, then the API bean must be included in the code. Once this has been done, it is possible to access the API methods. This can be accomplished using the following JSP tag:

```jsp
<jsp:useBean id="bizSite"
            class="com.savvion.sbm.bpmportal.bizsite.api.BizSiteBean" scope="session"/>
```

You need to properly specify the scope of the bean. In this case the scope is "session" and it is valid over multiple HTTP requests. "Request" might be used if it was desired that the bean remain valid only within the context of the current request. A more elaborate example follows.

**Example of API use within a JSP**

This example uses a variety of API methods to display data about the current available applications and tasks for the user.

```jsp
<title> BizSite API Example JSP </title>
<p><font face="Arial, Helvetica, sans-serif">
<b> BizSite API Example JSP </b></font></p>
<br>
<jsp:useBean id="bizSite"
            class="com.savvion.sbm.bpmportal.bizsite.api.BizSiteBean" scope="request"/>

<!-- DISPLAY THE SERVER'S TIME -->
<%= bizSite.getTime().toString() %><br><br>
<!-- LOG IN AS USER 'EBMS' -->
<b>Logging in:&nbsp;</b><br>
<%= bizSite.login( "ebms", "ebms", request ) %><br><br>
<!-- GET INFORMATION ABOUT THE CURRENT USER AND SESSION -->
<b>The current user is:&nbsp;</b>
```
<%= bizSite.getCurrentUser() %><br>

String[] groups = bizSite.getGroups();
for( int i=0; i < groups.length; i++ ) {

    &nbsp;&nbsp;&nbsp;&nbsp;<%= groups[i] %><br>
}

&nbsp;&nbsp;&nbsp;&nbsp;session:<i>/</i>&nbsp;&nbsp;
&nbsp;&nbsp;&nbsp;&nbsp;getSession()<br>
&nbsp;&nbsp;&nbsp;&nbsp;getRequestedSessionId()<br>
</hr>

<!-- APPLICATION INFORMATION -->
<br>
<br>
<b><u>APPLICATIONS</u></b><br>
<br>
%(com.savvion.sbm.bpmportal.bizsite.api.BizSiteApp[]) p;<br>
for ( int i = 0; i < p.length; i++ ) {

    &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;Application ( <%= i %> ) is:&nbsp;&nbsp;
    &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&n...
<b>Task ( <%= j %> ) is:&nbsp;</b>

&nbsp;&nbsp;<i>application:</i>&nbsp;&nbsp;<%= t[j].getAppName() %><br>

&nbsp;&nbsp;<i>creator:</i>&nbsp;&nbsp;<%= t[j].getCreator() %><br>

&nbsp;&nbsp;<i>priority:</i>&nbsp;&nbsp;<%= t[j].getPriority() %><br>

&nbsp;&nbsp;<i>start date:</i>&nbsp;&nbsp;<%= t[j].getStartDate() %><br>

&nbsp;&nbsp;<i>end date:</i>&nbsp;&nbsp;<%= t[j].getEndDate() %><br>

&nbsp;&nbsp;<i>assignee:</i>&nbsp;&nbsp;<%= t[j].getAssignee() %><br>

&nbsp;&nbsp;<i>dataslots:&nbsp;(sorted by name)</i><br>

<%com.savvion.sbm.bpmportal.bizsite.api.BizSiteDataslot[] slots; slots = t[j].getDataslotsSorted( "name" ); for( int k=0; k < slots.length; k++ ) {
  &nbsp;&nbsp;&nbsp;&nbsp;%= k %&gt;&nbsp;
  &nbsp;&nbsp;&nbsp;&nbsp;&lt;%= slots[k].getName() %&gt;&nbsp;
  &nbsp;&nbsp;&nbsp;&nbsp;(&lt;%= slots[k].getType() %&gt;)
  &nbsp;&nbsp;&nbsp;&nbsp;:&nbsp;&lt;%= (java.lang.String) slots[k].getValue() %&gt;
} %>

<!-- LOG OUT -->

Logging out:&nbsp;&nbsp;&lt;%= bizSite.logout() %&gt;

<br>

**Servlets**

Similarly, for servlets to make use of the API, the main API bean must be accessible from the servlet. You need to include the following package:

```java
com.savvion.sbm.bpmportal.bizsite.api.*;
```

To use the API, create an instance of BizSiteBean.

**Example of API use within a servlet**

```java
import com.savvion.sbm.bpmportal.bizsite.api.*;
...
... bizSiteBean bizSite = new BizSiteBean(); if( bizSite.login( "ebms", "ebms", request ) ) {
 ... ...
```
Customizing Login pages using BPM Sign-on API

This chapter describes an example of how a single Login page can be shared across multiple domains using the BPM Sign-on API and filters. For details, see the following topics:

- Reviewing requirements for a single Login page
- Reviewing logging in actions
- Reviewing logging out actions
- Reviewing session expiry scenarios
- Implementing filters for a single Login page
- DomainFilter
- BPM Sign-on API Filter
- Deploying BPM Sign-on API as the single Login page
- BPM Sign-on API Login page
- BPM Sign-on API Logout page
- Logging in with BPM Sign-on API
- Adding security with BPM Sign-on API
- Logging out with BPM Sign-on API
Reviewing requirements for a single Login page

The requirement is to use a single Login page to authenticate access to multiple separate Web sites. For example, if there is a Login page that provides access to three separate Web sites and a user tries to access site01, then the user is taken to a common Login page. After being authenticated, the user can also access site02 and site03, without being required to log in for each domain.

The following figure shows the expected user behavior.

**Figure 4: Single sign-on for multiple domains**

Machine1/Webapp1

Machine2/Webapp2

Machine3/Webapp3

Portal1

Business Process Portal

Portal2

Machine0/*/*/BPM Sign-on API

BPM Sign-on API

Database

User Manager

The main constraint is that the various domains need to be deployed as Web applications running inside containers that are compliant to Servlet 2.3 or above.

Reviewing logging in actions

In the example shown in Figure 4 on page 40, there are three separate machines/Web applications. The Machine2/WebApp2 is the Business Process Server machine. As an example, the following actions occur when BPM Sign-on is used to enable users to log in to multiple Web sites that share a common Login page.

- The first time the user tries to access any of the three Web applications, for example, WebApp2 (Business Process Server), BizPassFilter2 verifies that there is no valid session between the browser and WebApp2. As a result, the user is redirected to the common Login page residing...
under the BPM Sign-on domain on machine0. The link to the requested page is also communicated to BPM Sign-on as part of a redirection parameter, and is stored there for future references.

- After successful authentication, a valid BPM Sign-on session (session0) is maintained between the browser and the BPM Sign-on Web application. Then the user is redirected to the originally requested page. User ID and password are posted as part of the HTTP request.

- BizPassFilter2 in the Business Process Server domain gets the user ID and password to create the Business Process Server portal-specific beans (BizSiteBean, BPMManageBean, and so on) and put them into the newly created session, session2.

- The next time the user accesses Business Process Server functionality, it is possible to directly jump to the requested page, without entering User ID and password again, as long as a valid session (session2) is maintained between the browser and WebApp2. Also note that the necessary Business Process Server beans have already been created.

- Now if the user wants to access WebApp1, BizPassFilter1 does not recognize a valid session between the browser and WebApp1 at first, and redirects the user to the BPM Sign-on domain along with the WebApp1 information. Since session0 still exists, the login process is skipped and the browser is redirected back to WebApp1, which can now access the user ID and password passed through the HTTP request.

- In case the user clicks again on any link under WebApp1 after navigating to some other Web site, the corresponding page will be automatically brought up into the browser, because the bean initialization (in session1) steps have already taken place in the past.

- The BizPassFilter1 must initialize the session1 with WebApp1, which may require the user ID and password to create the necessary resources such as portal-specific beans.

- This model of single sign-on can be generalized to any number of Web applications.

### Reviewing logging out actions

When the user clicks on the logout button, not only the current session, but also all the active sessions belonging to BPM Sign-on (session0) and the other domains must be invalidated. For example, if all the sessions are active, and the current session (session2) gets a logout request, then session0, session1, and session3 must also be invalidated.

### Reviewing session expiry scenarios

An HTTP session maintained between the browser and a domain can expire after a certain amount of idle time. There are three possible scenarios:

- **Scenario 1**: The session with WebApp<i> has expired, but not BPM Sign-on session0. In that case, the user is still considered logged in. If the user tries to access the pages under WebApp<i>, session<i> is transparently created.

- **Scenario 2**: The session0 with BPM Sign-on domain has timed-out, but not session<i>. The user can still navigate within the WebApp<i> domain. However, when the user tries to access protected WebApps whose sessions have timed out, the third scenario needs to be considered.
Scenario 3: The session0 and session<i> are invalid. Then the user is considered to be logged out from the global domain, and the next time he tries to access another link, he will be directed to the single sign-on Login page.

Implementing filters for a single Login page

Considering all the requirements explained above, the following filters are implemented.

DomainFilter

This filter is used to protect accesses to any Web application.

- DomainFilter reads the context parameters BizPassLoginURL, which specifies the link to the common Login page (BPM Sign-on API Login page), and domainLogoutURL (the URL to request a logout action from the domain).
- It first checks whether the current session is valid or not by invoking the isSessionValid() method, which verifies the presence of domain-specific beans in the HTTP session.
- If the session is considered valid, then the requested page is displayed.
- Finally, in case everything else fails, the user is redirected to BPM Sign-on API Login page, along with the originally requested URL (the http parameter in domainTargetURL) and the URL to request a logout from the domain (the http parameter in domainLogoutURL).

BPM Sign-on API Filter

This is the filter deployed in the BPM Sign-on domain for guarding the Login page.

- BizPassFilter reads the context parameter BizPassHomeURL, which specifies the link that redirects to the home page, after a user directly accesses BPM Sign-on Login page.
- The domainTargetURL parameter is first retrieved from the request. If the parameter is not specified, the default value is taken as BizPassHomeURL. Then an HTTP session is created, and the domainTargetURL parameter value is mapped to the domainTargetURL session attribute.
- The domainLogoutURL parameter value needs to be added to the list of URLs (domainLogoutURLs session attribute) specifying the logout actions from the accessed domains protected by BPM Sign-on, to execute the global logout mechanism later on.
- The next time the BizPassFilter is invoked, the session creation may be skipped if a BPM Sign-on API session is detected, in which case the user is automatically redirected to the requested URL (domainTargetURL http parameter) using a Post method (BizPassUtilities.sendPostRedirect()), along with the user ID and password.
Deploying BPM Sign-on API as the single Login page

BPM Sign-on API resources are found in the `bpmsignon.jar` file under `<AppServer_Home>\webapps\deploy\sbm.war\WEB-INF\lib`.

BPM Sign-on API Login page

If you decide to use BPM Sign-on API as an entry point for all your Web sites, the default implementation of the Login page can be found under `<AppServer_Home>\webapps\deploy\sbm.war\bpmportal\login.jsp`. The `login.jsp` is the same as Business Process Portal Login page.

BPM Sign-on API Logout page

The BPM Sign-on API logout mechanism is encapsulated within the `logout.jsp` file under the `<AppServer_Home>\sbm\bpmportal` directory. Before invalidating the BPM Sign-on API session, you need to send a logout request to each domain protected by BPM Sign-on API and whose session is maintained with the current user.

Logging in with BPM Sign-on API

The scenario considered here is where BPM Sign-on is used as the common Login page.

1. The `bpmsignon.jar` file is located in the `<AppServer_Home>\sbm\WEB-INF\lib` directory.
2. For each domain to be protected by BPM Sign-on API, you need to deploy a filter that will validate the current session and redirect to BPM Sign-on Login page if necessary. Your filter class needs to extend `DomainFilter` and override the following methods:
   - `Protected boolean isSessionValid(HttpServletRequest request, HttpServletResponse response)`: checks whether the current session is valid or not. Similar to SBMFilter, you can verify the existence of session attributes or beans specific to your application.
   - `Protected void createValidSession(HttpServletRequest request, HttpServletResponse response)`: this method creates a valid session according to your application needs.
3. To compile your filter, you need to add `bpmsignon.jar` in your classpath.
4. When you deploy your filter class, you can either copy it under the `WEB-INF\classes` directory or package it as a `.jar` file under `WEB-INF\lib` of your Web application, along with `bpmsignon.jar`. 
5. To register your filter as part of your domain, you need to open `WEB-INF/web.xml` and edit the following deployment descriptors:

```xml
<context-param>
  <param-name>BizPassLoginURL</param-name>
  <param-value>http://peru:8080/bizpass/login/login.jsp</param-value>
</context-param>
<context-param>
  <param-name>domainLogoutURL</param-name>
  <param-value>http://peru:8080/yahoo/jsp/logout.jsp</param-value>
</context-param>
<filter>
  <filter-name>YahooFilter</filter-name>
  <filter-class>com.apps.bizpass.YahooFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>YahooFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

6. You need to provide the link to the BPM Sign-on Login page (in absolute path) as a context parameter (`BizPassLoginURL`). You also need to specify the `domainLogoutURL` context parameter, which indicates the URL to request a logout action from the domain.

You can give any name to your filter (`<filter-name>` element), but make sure that you reuse the same name inside the `<filter-mapping>` block. The `<filter-class>` element specifies the fully qualified name of the class implementing your filter.

The wildcard expression enclosed in the `<url-pattern>` element indicates that BPM Sign-on should protect every URL link under the Web application and that the user should be redirected to the common Login page if the session is invalid. You may decide to restrict the single sign-on watchdog functionality to a subset of JSP pages or servlets instead.

7. If you decide to use BPM Sign-on as an entry point for all your Web sites, you can customize the default `login.jsp` implementation in the `<App server specific folder>\sbm\bpmportal directory.`

8. Create `<BizPassWebApp>\login\login.jsp` directory or create your own Login page. However, you need to follow these guidelines:

   - Submit the user ID and password according to the HTTP parameters `BizPassUserID` and `BizPassUserPassword` respectively.
   - The `BizPassHomeURL` parameter should point to the default home page after a successful authentication. You also need the logout URL information corresponding to the default domain (domainLogoutURL context parameter). Finally, the BizPassFilter `<url-pattern>` in the `web.xml` of BPM Sign-on domain should specify the link to the Login page.

9. Edit the `WEB-INF\web.xml` deployment descriptor:

```xml
<context-param>
  <param-name>BizPassHomeURL</param-name>
  <param-value>http://peru:8080/yahoo/jsp/home.jsp</param-value>
</context-param>
<context-param>
  <param-name>domainLogoutURL</param-name>
  <param-value>http://peru:8080/yahoo/jsp/logout.jsp</param-value>
</context-param>
<filter>
  <filter-name>BizPassFilter</filter-name>
  <filter-class>com.apps.bizpass.BizPassFilter</filter-class>
</filter>
```
Adding security with BPM Sign-on API

A Web (formerly BPM WebFlow) Application can be accessed by typing in its URL in the Address bar of your browser. You can use BPM Sign-on API to design a Login page that will provide an additional level of security to accessing your Web applications, as shown in the following procedures.

To add security with BPM Sign-on API:

1. Open the web.xml file under the \web-inf directory of your application server.
2. Go to Filter Mappings > BizPassFilter <url-pattern> and define a deployment descriptor to create a Login page for a specific Web application — in this case, BS_Approval_Ver2, as shown in the following sample.

   <filter-mapping>
     <filter-name>BizPassFilter</filter-name>
     <url-pattern>/BPMWebFlow/BS_Approval_Ver2/*</url-pattern>
   </filter-mapping>

3. To create a Login page for all your Web Applications, define the following deployment descriptors.

   <filter-mapping>
     <filter-name>BizPassFilter</filter-name>
     <url-pattern>/BPMWebFlow/*</url-pattern>
   </filter-mapping>

4. The * symbol in BizPassFilter can be used to add a Login page to protect any Business Process Server resource.
   For example, to send users to a Login page before accessing all BP Server applications, use the following deployment descriptor.

   <filter-mapping>
     <filter-name>BizPassFilter</filter-name>
     <url-pattern>/ebmsapps/*</url-pattern>
   </filter-mapping>

Logging out with BPM Sign-on

When you logout, Business Process Portal calls the `session.invalidate()` method. It calls back the session unbound listener, registered while logging in. Business Process Portal then takes the necessary steps to clean up the allocated resources. It is recommended that a similar approach should be taken while customizing the logout page. Your customized BPM Sign-on Logout page may perform various operations as per your requirements, but it must call the `session.invalidate()` method.
Error handling

If an exception occurs during the execution of the domain filter (BP Server is not running while trying to instantiate a BizSite bean, for example), you can display a graceful warning message by overriding the handleFilterException() method inherited from DomainFilter.

By default, the handleFilterException() prints the error stack trace to the standard output and wraps the original exception into a ServletException.

Sample of a customized Login page

The following sample customization is designed for an organization that already has a portal/SSO capability and wants to integrate with Business Process Portal. Since the user has logged in with the organization, it does not want the user to be required to log in a second time for Business Process Portal. The sample below contains a section where the organization can perform its custom lookup for the user name and password. This customization is not visible to the user, and will transparently submit the login information to the BPM Sign-on API servlet filter, log the user in, and then redirect to the user’s intended target page.

Sample of a customized login page

```java
<%@ page contentType="text/html;charset=UTF-8" %>
<%@ page errorPage="login_error.jsp" %>
<% String _userID = "";
String _userPassword = "";
//Insert custom code for getting the user id and password here.
//_userID = <some custom code>;
//_userPassword = <some other custom code>;
String targetURL = request.getParameter("domainTargetURL");
// You can change this value to whatever default page you want to go to.
// Currently it is set to go to the Home->My Tasks page.
if (targetURL == null)
targetURL = "myhome/bizsite.task";
//This code checks the target url for any HTTP parameters.
//By default the browser will ignore any HTTP parameters included in the action attribute of the form.
//If found it will insert them into the page as a hidden parameter so the form will pick them up.
String paramName=null;
String paramValue=null;
if ((targetURL.indexOf("?",:) > 0) && (targetURL.indexOf("=",:) > 0))
{
    paramName=targetURL.substring(targetURL.indexOf("?")+,targetURL.indexOf("=")+1);
    paramValue=targetURL.substring(targetURL.indexOf("=")+1);
}
```
Sample of a customized Login page

```html
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html >
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
<table border="0" cellpadding="0" cellspacing="0" width="100%">
<tr>
  <td>
    <form name="login" action="${targetURL}" method="post">
      <input type="hidden" name="BizPassUserID" value="${_userID}">
      <input type="hidden" name="BizPassUserPassword" value="${_userPassword}">
    </form>
    <script language="javascript">
    <!--
        document.login.submit();
    -->
    </script>
  </td>
</tr>
</table>
</body>
</html>
```
Customizing User Management Java interface

This chapter describes the User Management Java interface and how to customize it. For details, see the following topics:

- The User Management Java interface
- Customizing the userManager interface
- LDAP realm customization
- OEHybrid realm customization
- Manipulating customized User Management

The User Management Java interface

User Management provides the following interfaces and user realms, four Java interfaces: User, Group, Realm, and Advanced Group and two realms: LDAP and JDBC

**Note:** If you want to know more about LDAP and Business Process Portal customization, or if you have any other specific customization requirement which is not covered in the documentation, please contact OpenEdge Product Support. For more information, see Product support contact information on page 12.
Customizing the userManager interface

If you want to implement, custom realm, perform the following:

1. Provide your custom realm classes in this umext.jar file.
   
   This is an empty jar file available with BP Server installation.
   
   This file is available at two places: OEBPS_HOME\lib and WEBAPP_HOME\web-inf\lib folder.
   
   **Important:** The umext.jar with custom realm classes should be provided in both the locations mentioned above.

2. Provide the following configuration parameter changes in umacl.conf file.
   
   a) Specify Custom for usermgr.realm.type value.
   
   b) Provide fully qualified custom implementation class name for usermgr.realm.provider value.

   The following table lists the parameter values you have to set in umacl.conf file for customizing the user manager interface.

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Parameter value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>usermgr.realm.type</td>
<td>'jdbc' or 'ldap' or 'ldaphybrid' or 'custom'</td>
<td>Realm type</td>
</tr>
<tr>
<td>usermgr.realm.provider</td>
<td>Fully qualified class name of the realm implementation class, if the usermgr.realm.type parameter is set to 'custom'. If the usermgr.realm.type parameter is set to any value other than 'custom', then this parameter value is ignored.</td>
<td>User realm implementation</td>
</tr>
</tbody>
</table>

LDAP realm customization

The LDAP Realm is developed based on the default schema in the Sun Java System Directory server, and the Microsoft Active Directory server. However, some LDAP users must define their own schemas to meet their particular requirements. To make these non-standard schemas operate with Business Process Server, a customization mechanism is provided in LDAP Realm to recognize them.

Business Process Server stores the description of the non-standard schema in umacl.conf located in the OEBPS_HOME\conf directory. Although each LDAP server product distinguishes itself by a set of predefined schema, most LDAP server products are flexible enough to allow users to define their own schema. For parameters not set in umacl.conf, the default values will be used based on the LDAP server type.
OEHybrid realm customization

OpenEdge single point of authentication (SPA) is an authentication service for use in realm-based systems. Business Process Server (BP Server) now supports the single point of authentication service using the OEHybrid realm. OEHybrid realm is a combination of JDBC realm and OpenEdge AppServer based service. BP Server supports custom realm, which is how the OEHybrid realm single point of authentication feature is presented.

For more information on configuration OEHybrid realm for single point of authentication, see the Business Process Portal Administrator's Guide.

Manipulating customized User Management

The Administration Module of Business Process Portal contains pages that can manipulate the UserManagement data by calling UserManagement API. Three realm implementations are available in Business Process Server: JDBC, LDAP, and OEHybrid realm. According to the properties of implementation, the Administration Module displays various pages to the realms.

Table 9: Manipulating User Management

<table>
<thead>
<tr>
<th>Capabilities</th>
<th>JDBC realm</th>
<th>LDAP realm</th>
<th>OE realm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search User</td>
<td>Page displays</td>
<td>Page displays</td>
<td>Page displays</td>
</tr>
<tr>
<td>Create User/Group</td>
<td>Page displays</td>
<td>N/A</td>
<td>Page displays³</td>
</tr>
<tr>
<td>Remove User/Group</td>
<td>Page displays</td>
<td>N/A</td>
<td>Page displays</td>
</tr>
<tr>
<td>Update User</td>
<td>Page displays</td>
<td>Page displays⁴</td>
<td>Page displays</td>
</tr>
<tr>
<td>Add group member</td>
<td>Page displays</td>
<td>N/A</td>
<td>Page displays</td>
</tr>
<tr>
<td>Remove group member</td>
<td>Page displays</td>
<td>N/A</td>
<td>Page displays</td>
</tr>
<tr>
<td>Change Role</td>
<td>Page displays</td>
<td>Page displays</td>
<td>Page displays</td>
</tr>
</tbody>
</table>

The Administration Module recognizes the realm by the last part of the name. For example, the Administration Module considers the realm name that ends with "JDBCRealm" as a type of JDBC realm and the Administration Module displays the pages similar JDBCRealm. Therefore, if your implementation has properties similar to JDBCRealm, you could name your realm class that ended with "JDBCRealm," such as "com.myCom.usermanage.myJDBCRealm" or simply "myJDBCRealm."

³ For OE realm, the Create User button is disabled.
⁴ For LDAP realm, some user information (last name, first name, email and phone number) are stored in the LDAP server. You cannot update this information in the Administration Module.
Customizing the Business Process Portal

Business Process Portal is the cross-over portal for Business Process Server Administrators, Managers, Application Developers, and Users. As a unified, customizable portal for accessing all Business Process Server functionality, Business Process Portal allows all user types to access all features of Business Process Server to which they are granted permission to access. For details, see the following topics:

- Business Process Portal features
- BPM Single Sign-on API
- User Manager and Access Control Management
- User permissions
- The Portal APIs
- Portal customization
- The Login page
- Login Error page
- Using the Shortcut keys
- Adding a new shortcut
- Modifying an existing shortcut
- Hiding the Menu bar
- Customizing a workstep to use concurrent processing
Business Process Portal features

The key features of Business Process Portal are:

- The user authentication process (single sign-on with BPM Sign-on API).
- The Business Process Portal menu and its interaction with the User Manager and ACLManager.
- Implementation of the Business Process Portal on the Portal API.

BPM Single Sign-on API

Single sign-on architecture provides the following:

- Single point of authentication for access to all Business Process Server features.
- Intercepts unauthorized access to Business Process Server features.

The single sign-on capability provides portal users access to Business Process Server features, based on their access permission.

Business Process Server allows BPM Sign-on to add single sign-on capability to Business Process Server portals. BPM Sign-on API is a servlet that:

- Intercepts a portal user's request to login to Business Process Server.
- Authenticates users, if they have not been authenticated already.
- Has the ability to redirect to a previously requested resource.

To login to Business Process Portal:

Enter the following URL in your browser, http://<host_name>:<port_number>/sbm/bpmportal/login.jsp

The Login page is displayed as shown below:

Figure 5: Business Process Portal Login page
The file, login.jsp, is a customizable login page which prompts users to enter their user name and password. This information goes to the BPM Sign-on API Servlet which then creates a session and redirects the user to their Home page. If a session is declared invalid due to a time out, then the user is redirected to the Business Process Portal Login page and may resume after providing a valid user name and password.

**Note:** The Business Process Portal session timeout is controlled by the application server. Please refer to the application server documentation for information on how to modify the session timeout value.

---

### User Manager and Access Control Management

Business Process Portal quickly and efficiently retrieves all necessary information required to prepare a user’s **Home** page. The access rights and permissions to users and groups are granted in the Administration module’s Access Control Management (ACLManager). The required information about each user’s permissions is cached with the User Manager and retrieved during user authentication, thereby eliminating calls to the ACLManager to retrieve a user’s permission list. Refer to the *Business Process Portal Administrator’s Guide*, chapter on Access Control Management for more details.

### User permissions

By default, all users have permission to access the **Home** module, and its menu. The Business Process Server Administrator must grant a user permission to access other modules such as Management, Administration, and any other customized modules that may be added.

The following figure shows the **Home** page for user "admin" with the management permissions. When "admin" clicks the Management tab, the Management menu is displayed. Similar behavior is seen for other tabs.

### The Portal APIs

The Home Module is implemented entirely on the BizSite bean and BPM CustomUI API. The Management module is implemented on the BPM Manage bean and BPM Manage API.

### Portal customization

You may customize a set of default jsp forms for integration with corporate single sign-on portal, the **Login** page and the plug-ins to share sessions. You can customize these to match the style or needs of your organization. This section explains how.
For localization issues pertaining to these pages, refer to the Configuring the Home Module in the Business Process Portal Administrator's Guide.

The Login page

With a text or HTML editor that supports JSP, you can customize the login.jsp file. The file is stored under the <App server specific folder>/sbm/bpmportal directory. The JSP code used to customize the Login page is provided in the following section.

Note: If you wish to customize the interface in the Business Process Server Home page, remember to include the path of the BPM Sign-on API servlet in your code.

JSP code for customizing the login page

```jsp
<%@ page contentType="text/html;charset=UTF-8" pageEncoding="UTF-8" %>
<%@ page errorPage="login_error.jsp" %>
<%@ page import="java.util.*" %>
<%@ page import="com.savvion.sbm.bizmanage.api.*" %>
<%@ page import="com.savvion.sbm.bpmportal.util.PortalConstants" %>
<%@ page import="com.savvion.sbm.bizmanage.pagegenerator.ServletTools" %>
<%@ taglib uri="http://java.sun.com/jstl/fmt" prefix="fmt" %>
<%@ taglib uri="http://java.sun.com/jstl/core" prefix="c" %>
<%@ taglib uri="/bpmportal/tld/bpmportal.tld" prefix="sbm" %>
<jsp:useBean id="bizManage" class="com.savvion.sbm.bizmanage.api.BizManageBean" scope="session" />
<%@ taglib uri="http://java.sun.com/jstl/fmt" prefix="fmt" %>
<%@ taglib uri="http://java.sun.com/jstl/core" prefix="c" %>
<%@ taglib uri="/bpmportal/tld/bpmportal.tld" prefix="sbm" %>
<jsp:useBean id="bizManage" class="com.savvion.sbm.bizmanage.api.BizManageBean" scope="session" />
<%xml version="1.0" encoding="UTF-8"%>
<!DOCTYPE html >
<html xmlns="http://www.w3.org/1999/xhtml">
<% String base = ServletTools.getBaseLocation(request);
bizManage.setRequest(request);
bizManage.setResponse(response);
bizManage.init(config.getServletContext());
String paramName = null;
String paramValue = null;
String targetURL = request.getParameter(PortalConstants.DOMAIN_TARGET_URL);
if (targetURL == null)
    targetURL = "myhome/redirect";
targetURL = base + targetURL;
if ((targetURL.indexOf("?") > 0) && (targetURL.indexOf("=") > 0))
{ paramName = targetURL.substring(targetURL.indexOf("=")+1, targetURL.indexOf("="));
    paramValue = targetURL.substring(targetURL.indexOf("=")+1);
}
String requestFrom = (String) request.getAttribute(PortalConstants.REQUEST_FROM);
String errorMsg = request.getParameter(PortalConstants.BIZPASS_ERROR_MSG);
<% %>
</head>
<sbm:setLocale value="<%= bizManage.getLocale() %>">

<sbm:setBundle basename="properties/bpmportal" scope="page" />
<title><sbm:message key="Login" /> - <sbm:message key="SBM" /></title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<jsp:include page="common/include_css.jsp">
    <jsp:param name="login" value="true" />
</jsp:include>
<script type="text/javascript" src="<%= base %>/javascript/top.js"></script>
<script type="text/javascript" src="<%= base %>/javascript/adapter/ext/ext-base.js"></script>
<script type="text/javascript" src="<%= base %>/javascript/ext/ext-all.js"></script>
<script type="text/javascript" src="<%= base %>/javascript/bmencryptutils.js"></script>
</head>
<body>
</body>
</html>
```
function checkformvals(theForm) {
    if (theForm.<%= PortalConstants.BIZPASS_USER_ID%>.value == '') {
        alert("<sbm:message key="username" />");
        return false;
    } else {
        var uIdTrimLen = trimString(theForm.<%= PortalConstants.BIZPASS_USER_ID%>.value).length;
        var uIdLen = theForm.<%= PortalConstants.BIZPASS_USER_ID%>.value.length;
        if (uIdTrimLen != uIdLen) {
            alert("<sbm:message key="UserBlankSpaceValidation" />");
            return false;
        }
    }
    if (theForm.<%= PortalConstants.BIZPASS_USER_PASSWORD%>.value == '') {
        alert("<sbm:message key="password" />");
        return false;
    }
    return true;
}

function setTime() {
    var today = new Date();
    document.login.timeZone.value = today.getTimezoneOffset();
}

function doEncrypt() {
    var isPswrdEncrypted = document.getElementById("<%= PortalConstants.IS_PASSWORD_ENCRYPTED %>").value;
    if (isPswrdEncrypted == "true") {
        try {
            var userEl = document.getElementById("txtBizPassUserID");
            var pwdEl = document.getElementById("txtBizPassUserPassword");
            pwdEl.value = Bm.BmEncryptUtils.encrypt(userEl.value, pwdEl.value, ";" + PortalConstants.DASH_SEPERATOR + ");
            return true;
        } catch (error) {
            throw error;
        }
    } else {
        return true;
    }
} //-->
Chapter 7: Customizing the Business Process Portal

<% if(PortalConstants.LOGOUT.equals(requestFrom)) {
    <sbm:message key="logoutMessage1" /><br/><sbm:message key="logoutMessage2" />
} else if(PortalConstants.LOGIN_ERROR.equals(requestFrom)) {
    if(request.getParameter(PortalConstants.BIZPASS_ERROR_MSG).contains("BM_PW16100")) {
        <sbm:message key="InvalidUserNameAlert" />
    } else {
        <%=errorMsg%>
    }
} else if(PortalConstants.XSRF.equals(requestFrom)) {
    <sbm:message key="XSRFAttackObserved" />
} else {
    <sbm:message key="EnterLogin" />
}

<form name="LoginForm" method="post" action="logout" onsubmit="return validateForm()">
    <table width="100%" border="0" cellspacing="0" cellpadding="0">
        <tr><td><input type="text" name="<%= PortalConstants.BIZPASS_USER_ID %>>" id="txtBizPassUserID" onFocus="setTime()" class="InptTxt" size="32" maxlength="255" tabindex="1"></input>
        <label for="txtBizPassUserID"><sbm:message key="username"></label></td>
    </tr>
    <tr><td><input type="password" name="<%= PortalConstants.BIZPASS_USER_PASSWORD %>>" id="txtBizPassUserPassword" onFocus="setTime()" class="InptTxt" size="32" maxlength="255" tabindex="2"></input>
        <label for="txtBizPassUserPassword"><sbm:message key="password"></label></td>
    </tr>
</form>
Login Error page

When the end user fails to provide the proper ID or password while attempting to login, the failed authentication page appears. You may localize and customize this file, login_error.jsp located in the folder <App server specific folder>\sbm\bpmportal.

Using the Shortcut keys

For your convenience, Business Process Portal provides keyboard shortcuts to access its various module menus and menu options. However, you can add a new shortcut key or change an existing shortcut key as per your requirement. You can do this by editing the $webapp\sbm\bpmportal\common\include_menu_static.jsp file.

Note: To reflect your changes, make sure to clear your cache after modifying the javascript file. For more details, refer to your browser’s documentation.
Adding a new shortcut

If you want to add a new shortcut key, then do the following:

1. Open the $webapp\sbm\bpmportal\common\include_menu_static.jsp file.
2. Add a new variable in the list of variables for keyboard shortcuts, above the function handleShortKeys(evt).
3. Assign that variable a value equivalent to the ASCII value of the key combination.
4. Add a new case statement checking for the value of the new variable in the switch statement of function handleShortKeys(evt). Below the case, specify one of the predefined methods to be used if the case statement is evaluated as True.
5. Save the file.

Modifying an existing shortcut

To modify an existing shortcut:

1. If you want to assign an existing shortcut to perform a different action, then modify the requisite case of the switch statement in the function handleShortKeys(evt). It is recommended that you should use the predefined methods defined in the include_menu_static.jsp file for pointing to the URLs.

   The following example illustrates how to change the Ctrl+Shift+T key to open the Links page instead of the Task List page.

   ```javascript
   switch(keyAsciiCode)
   {
   case SHIFT_CNTR_T:
     goToTasks();
     break;
   }
   ```

2. Replace the goToTasks() method in the above example with the goToLinks() method to open the Links page.

   ```javascript
   switch(keyAsciiCode)
   {
   case SHIFT_CNTR_T:
     goToLinks();
     break;
   }
   ```

Hiding the Menu bar

If you have integrated Business Process Portal with your corporate portal, and therefore, to utilize the screen "real-estate" effectively, you can hide the module menu bar of Business Process Portal screens. To do so, you need to modify the jsp file of the screen for which you want to hide the menu bar. Note that, if you want to hide the menu bar for the Task List page, you will have to modify the myhome\tasks.jsp and myhome\taskframe.jsp files from the $webapp\sbm\bpmportal directory. However, for other screens, you need to modify only the corresponding jsp file.
Customizing a workstep to use concurrent processing

If you want to create a custom JSP involving redirection, linking, or using AJAX calls, then to enable concurrency, you must pass the fiid parameter as a part of all URL queries or POST requests. You can obtain the value of this parameter by using the call request.getParameter("fiid"). OpenEdge Business Process Server provides the following API to this parameter in your queries.

public static String makeSessionURL(String url, String fid);

The API is automatically added to the JSP import list as shown in the following example.

<a href="<%=VirtualSessionManager.makeSessionURL(MyCustomJump, request.getParameter("fiid"))%>">Go to my activity</a>

When you generate the URL using JavaScript, the JavaScript must obtain the fiid parameter value as a part of JSP generated variable. For example:

<script> var globalfid = '<%= = request.getParameter("fiid")%>'; </script>

If a page contains the fiid value as a part of the form value, then you can retrieve and use it in JavaScript expressions as shown below:

url += '&fiid=' + encodeURIComponent(document. myTaskForm.fiid.value);
Alternatively, JavaScript can parse the request query, and extract the parameter value. If you add the value in URL using JavaScript, then its value must be `encodeURIComponent()`. You also must sanitize the JavaScript value using the utility method `SBMUtil.jsEncode()`.

If you use a custom-generated, task-access URL, then it must include the `fiid` parameter. The following rules are recommended while including the `fiid` parameter as task URLs are generated out of the scope of any existing flow ID, and every task must obtain a unique flow ID.

- If the URL contains a `bizsite_taskId` parameter, then prefix `wi` to the `fiid` value as shown in the following example.
  
  ```
  bizsite_taskId=3527&fiid=wi3527
  ```

- If the URL contains a `wiId` parameter, then prefix `wi` to the `fiid` value.
  
  ```
  wiId=3527&skip=false&fiid=wi3527
  ```

- If the URL contains a `cwiId` parameter, then prefix `ci` to the `fiid` value as shown in the following example.
  
  ```
  cwiId=3527&cenabled=false&skip=false&fiid=ci3527
  ```

It is recommended that you use a unique `fiid` value in the scope of a single user session.
Customizing the Business Process Portal look and feel

Business Process Server provides Business Process Portal with a default look and feel, which you can customize to meet the customer requirement. Quite often this need arises when an organization wishes for a look and feel consistent with its own corporate standards.

Business Process Portal uses themes to manage the look and feel. A theme provides visually consistent color, color gradients, background; and a related set of icons and buttons. The Business Process Server default theme is named theme01, which has a complementary application-specific theme called apptheme01.

The Task Details page, the Business Process Server Applications page and other pages in your applications use apptheme01 as the default application theme. The out-of-box Business Process Portal uses the same look and feel for theme01 and apptheme01. However, you can retain the default theme01, and use another theme for all application pages. The utilities such as calendar invoked from these two pages also use apptheme01.

While customizing the look and feel, you have two options:

- Retain the current theme and modify the existing stylesheets
- Design a new theme with modified stylesheets.

If you need to change only a few screen elements, modifying the existing stylesheets is an easier option. On the other hand, if you expect major changes in the look and feel, we recommend that you design a new theme.

For details, see the following topics:

- Screen layout
- Modifying stylesheets
Screen layout

All the web pages of Business Process Portal are designed with standard screen areas. The idea behind it is to separate the static and dynamic contents, and control the navigation correctly. The following figure shows the main elements in a typical page in Business Process Portal.

Note: The screens in this chapter are taken using a particular theme. Your Business Process Server look and feel will vary depending on the theme your administrator has set for your organization.

The screen areas are listed in the following table.
Table 10: Business Process Portal screen areas

<table>
<thead>
<tr>
<th>Screen Areas</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module Tab</td>
<td>Features related to specific module are grouped under Module Tab. Each Module Tab has its Module Menu.</td>
</tr>
<tr>
<td>Module Menu</td>
<td>Features related to specific module are listed under Module Tab as Module Menu.</td>
</tr>
<tr>
<td>Global Links</td>
<td>These links located in the top right corner of every page that are available at all times during the session.</td>
</tr>
<tr>
<td>Link Trail</td>
<td>Link Trail shows the pages you navigated through to reach the current page. You can also click any link in the link trail and go back to that page.</td>
</tr>
<tr>
<td>Workspace</td>
<td>Data is displayed in the Workspace. Data is either displayed as Forms or List.</td>
</tr>
<tr>
<td>Paging Control</td>
<td>The Paging Control is located immediately above and below the list, and allows navigation through the pages of the current list.</td>
</tr>
<tr>
<td>Filter Bar</td>
<td>This bar provides settings for filters so that the screen clutter is reduced and only the relevant data is displayed.</td>
</tr>
<tr>
<td>Search Bar</td>
<td>Search Bar allows specifying text to locate it in the list.</td>
</tr>
<tr>
<td>Command Button</td>
<td>The commands that help you interact with the data are represented by Command Buttons.</td>
</tr>
</tbody>
</table>

Modifying stylesheets

If you need to alter only a few screen elements, we recommend modifying the existing stylesheets of the default theme - `theme01`. The Task Details page, the Business Process Server Applications page and other pages in your applications use a different theme called `apptheme01`. The out of the box Business Process Portal uses the same look and feel for `theme01` and `apptheme01`. However, you can retain the default `theme01`, and use another theme for all application pages. The utilities such as calendar invoked from these two pages also use `apptheme01`.

When you are modifying the stylesheets of `theme01`, ensure that `apptheme01` is also correspondingly modified so as to maintain consistency.

The following sections describe various modules, attributes and variables you need to work on during the customization of the Business Process Portal.

The paths to css, buttons, icons and jsp files are specific to the application server.

Default CSS

Business Process Portal uses a number of stylesheets (.css files) listed in the folder:

`<App server specific folder>\sbm\bpmportal\css\theme01`
The main CSS style file is `bm-all.css`, which contains majority of application styles. The class names reflect the expected use. For example, class `LogoBg` is used to define style for logo background.

**Buttons**

The buttons use classes to define their style, and classes use images to define the look and feel. The classes are listed in the folder:

```
<App server specific folder>\sbm\bpmportal\css\theme01
```

and they use images listed in the folder:

```
<App server specific folder>\sbm\bpmportal\css\theme01\images
```

The image names reflect the expected use. For example, `scrn_button.gif` is used to show the normal state of screen button, `scrn_button_dis.gif` is used to show the disabled state, and `scrn_button_hvr.gif` is used to show the mouseover state.

If you replace any images for buttons, ensure that the original size is not changed.

**Icons**

The icons used in Business Process Portal are listed in the folder:

```
<App server specific folder>\sbm\bpmportal\css\theme01\images
```

The image names reflect the expected use. For example, `icon_support.gif` is used to show the support link icon.

If you replace any icons, ensure that the original size is not changed. Also verify that the icons displayed in the Business Process Portal are of the correct resolution.

**JSP files**

The individual pages of Business Process Portal and the popup windows are jsp files. The jsps files are located within their corresponding folders in the folder:

```
<App server specific folder>\sbm\bpmportal
```

The jsp file names reflect the expected use. For example, `tasks.jsp` and `task_list.jsp` are used to display list of tasks.

When you open a particular page, you can see the jsp file name in the browser address bar.

**Customization guidelines**

For any alteration in images, maintain their existing size in pixels. Any deviation will disturb the placement of other screen elements.

For any changes in fonts, maintain the existing size. Any deviation will disturb the placement of other screen elements.

Verify that the page contents fit 1024x768 pixels so that vertical and horizontal scroll are avoided.

Confirm that the new design is rendered correctly in the supported browsers.

Ensure that the documentation is updated to maintain consistency with the new screen layout.
Sample customization
In this case of customization, customer requirements are:

• Change the Business Process Server Logo to its corporate logo, and its background color.
• Change the color of Global Links.
• Change the look and feel of all buttons at the bottom of the page to the standard buttons used within the organization, intranet, and corporate website.

Default view
The following figure shows the default view for the Task List page.

Figure 7: Task List - Default view.

Modifying the files
You can do customization by modifying the files as explained in the following procedure.

1. Before changing the logo, view any page and open its source file to check for places where a logo is used (include_menu_static.jsp). Note all the instances.
2. To change the Business Process Server logo to a customer's corporate logo, open `<App server specific folder>`\`sbm\bpmportal\css\theme01\bm-all.css` file, and change the following style:

```
.logoImage {
    background: transparent url(images/bpmportal_logo_ie6.gif) no-repeat scroll top left;
    float: left;
    width: 390px;
    height: 32px;
    padding-top: 20px
}
```

3. Replace `bpmportal_logo_ie6.gif` with your company logo. For changing logo in the main login page, you need to replace `progress_savvion_busimanager_ie6.gif` file in images directory with your corporate logo.

Ensure that the background is consistent with the current settings.

**Note:** For any alteration in images, maintain their existing size in pixels. Any deviation will disturb the placement of other screen elements.

4. Go to, `<App server specific folder>`\`sbm\bpmportal\css\theme01` and locate and replace the following classes with the reference to corresponding image icons:

- supportIcon
- helpIcon
- aboutIcon
- logoutIcon

**Note:** For any changes in fonts, maintain the existing size. Any deviation will disturb the placement of other screen elements.

5. To change the style of the command buttons, first view any page and open its source file. In this example, Task List page is used. Search for the string "Complete", which is the button title in this page.

The classes associated with the Complete button are `ScrnButtonHover`, `ScrnButton`, and `ScrnButtonDis` in the `bm-all.css`.

6. Go to, `<App server specific folder>`\`sbm\bpmportal\css\theme01` and locate these classes:

- ScrnButton
- ScrnButton_dis
- ScrnButton_hover

These classes are located in `bm-all.css`. Edit as per your requirement.

Also change the `scrn_button.gif`, `scrn_button_dis.gif`, and `scrn_button_hvr.gif` images.
**Note:** In certain themes, the background color may have a gradient, and the buttons use images to define the background. In such cases, three different images will be associated with the button for three different states - normal, disabled, and on mouseover. You can locate these images in the subfolder `\images`, and edit as required.

---

## Customized view

The following figure shows the customized view for Task List. Note the new logo, new look of global links, and the new Complete button.

**Figure 8: Task List - Custom View.**

---

## Customizing themes

The Business Process Portal look and feel is set by using themes. A theme provides visually consistent color, color gradients, background; and a related set of icons and buttons. We recommend that you first design a theme conforming to the customer's corporate brand and then apply it to the Business Process Portal. However, it requires substantial efforts, and should be done only in consultation with Progress OpenEdge.

Designing a theme involves:

- Arriving at a template taking reference of the organization, its brand, internal style guide, intranet, and corporate website.

- Designing the required graphics, icons, and buttons.

- Testing with different browsers.
• Ensuring that the documentation is consistent with the new screen layout.

It is always a good idea to apply the theme to a few sample pages and then fine tune the design.

## Customizing AppThemes

You can use an apptheme to change the look and feel of presentation pages — that is, the Task Details page, the Business Process Server Applications page and other pages in your applications. To customize the apptheme, you must modify the classes in the `sbm_app01.css` file in the `apptheme01` directory. Changing the look and feel for an apptheme will modify all the pages in all your applications.

**To customize the look and feel of pages in your applications for the Default form or JSP generated using XSL interface options:**

1. Go to `<App server specific folder>\sbm\bpmportal\css\apptheme01\sbm_app01.css`.
2. Open `sbm_app01.css` with an external editor.
3. Make required changes in the appropriate classes. See the table below for information on apptheme classes.
4. Save the changes in the external editor you are using, and refresh.
5. The changes you have made are now displayed in the presentation pages of the applications.

Some of the more frequently used classes in the `sbm_app01.css` file are presented in the following table.

**Table 11: AppTheme classes**

<table>
<thead>
<tr>
<th>Screen Element</th>
<th>Class name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>.ApSegTitle</td>
<td>Controls the title of the page (HTML tag: Text).</td>
</tr>
<tr>
<td>Data Label</td>
<td>.ApSegDataLabel</td>
<td>Controls the labels of all user-defined dataslots (HTML tag: TD).</td>
</tr>
<tr>
<td>Data Value</td>
<td>.ApSegDataVal</td>
<td>Controls the content of all user-defined dataslots (HTML tag: TD).</td>
</tr>
<tr>
<td>General Label</td>
<td>.ApSegGenLabel</td>
<td>Controls the labels of all pre-defined system dataslots (HTML tag: TD).</td>
</tr>
<tr>
<td></td>
<td>.ApSegDataVal</td>
<td>Controls the content of all pre-defined system dataslots (HTML tag: TD).</td>
</tr>
</tbody>
</table>
**Note:** If you wish to change the images used in the `appteme01`, then you should place the new images in the `<App server specific folder>\sbm\bpmportal\css\appteme01\images directory.`
Customizing dashboard

This chapter explains you how to customize the dashboard in Business Process Portal. For details, see the following topics:

• Overview
• Configuring dashboards
• Configuring dashboard layout
• Configuring layout renderer
• Defining custom layouts
• Configuring a column-based layout renderer
• Configuring a non-column-based layout renderer
• Adding layout renderer to the dashboard configuration

Overview

Business Process Server provides the following layouts for Business Process Portal dashboard by default:

• One Column
• Two Columns 50-50
• Two Columns 30-70
However, these layouts may not fulfill all users’ requirements. There can be scenarios where you may want to have a ‘six-column’ or ‘n-column’ layout. For example, if a user has a larger or wider screen, he/she can accommodate more number of columns on his/her dashboard. To support such requirements, dashboard framework provides additional configuration options to define custom dashboard layouts. The out-of-the-box as well as custom layouts are available in the Layouts drop-down list while adding/editing a dashboard. This enables a user to add and use customized layout in the dashboard.

Configuring dashboards

A broad outline of steps to configure custom dashboards is configuring dashboard layout and layout render.

Configuring dashboard layout

You can define the additional layouts required for customization using dashboard-config.xml file from the OEBPS_HOME/conf folder. The XML schema file dashboard-config.xsd is located in the OEBPS_HOME/conf/resources/bpmportal folder. The dashboard-config.xml file has the following structure.

```xml
<Dashboard>
  <Layouts>
    <Layout id="31" type="column">
      <Name>FourColumns</Name>
      <Renderer>FourColumnsLayout.jsp</Renderer>
      <Description>This is a layout with four columns</Description>
      <Columns values="4" />
    </Layout>
    <Layout id="32" type="other">
      <Name>AnotherLayout</Name>
      <Renderer>AnotherLayout.jsp</Renderer>
      <Description>This is a test layout with no columns</Description>
    </Layout>
  </Layouts>
</Dashboard>
```

The following table explains the tags and their purposes for defining custom layout in a dashboard.

**Table 12: Policy and methods**

<table>
<thead>
<tr>
<th>Tag name</th>
<th>Description</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dashboard</td>
<td>Root tag for the XML</td>
<td>This tag should have one and only one &lt;Layouts&gt; tag.</td>
</tr>
</tbody>
</table>
### Configuring layout renderer

After you configure your dashboard layout, you need to configure the layout renderer. You need to configure the layout renderer differently for column-based dashboard layout and non-column-based dashboard layout.

<table>
<thead>
<tr>
<th>Tag name</th>
<th>Description</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layouts</td>
<td>This tag can contain multiple Layout tags.</td>
<td>This tag can have zero or more Layout tags.</td>
</tr>
<tr>
<td>Layout</td>
<td>This is the child tag of Layouts tag and has attributes like id and type.</td>
<td>‘id’ attribute is mandatory, and it should be between the range of 30-99 (both inclusive). ‘type’ attribute is mandatory, and the valid values for this tag are ‘column’ and ‘other’. Note: If ‘type’ attribute is set to ‘column’, then this will be treated as column layout dashboard and &lt;Columns&gt; tag will be mandatory. If the ‘type’ attribute is set to ‘other’, then this will be a non-column layout, and you will need to take care of placing the widgets in the layout renderer.</td>
</tr>
<tr>
<td>Name</td>
<td>This is the name of a layout, which appears in the Layout drop down list while creating/editing a dashboard.</td>
<td>The name can be maximum 60 characters in length.</td>
</tr>
<tr>
<td>Description</td>
<td>Description of the defined layout.</td>
<td>This is an optional field. The description can be maximum 255 characters in length.</td>
</tr>
<tr>
<td>Renderer</td>
<td>This is the name of the JSP file that is called when you select a layout.</td>
<td>This is a JSP file name. It should not contain ‘/’. The server searches for this filename in the &lt;web-context&gt;/bpmportal/layout/ext folder.</td>
</tr>
<tr>
<td>Columns</td>
<td>This tag is mandatory when the ‘type’ attribute of the ‘Layout’ tag is set to ‘column’. This tag has an attribute ‘values’. This represents the number of columns in the layout.</td>
<td>The ‘values’ attribute has a numeric value.</td>
</tr>
</tbody>
</table>

When you start the portal server, it reads the dashboard-config.xml file. However, when a user clicks ‘Edit Dashboard’ in Business Process Portal, it will refresh its memory contents.

The dashboard-config.xml file provides out-of-the-box configuration for four-column and five-column layouts. However, these layouts are commented in the XML file. You need to uncomment appropriate sections to enable these layouts. The default renderers for these layouts are FourColumnsLayout.jsp and FiveColumnsLayout.jsp respectively. The FourColumnsLayout.jsp file arranges the added widgets in four columns of equal widths. Similarly, the FiveColumnsLayout.jsp file arranges the added widgets in five columns of equal width.
Defining custom layouts

You can define custom layouts for the dashboard by configuring the `dashboard-config.xml` file. The custom layout can be column based or non-column based. Business Process Portal dashboard framework supports column based layouts in which the framework allows you to drag and drop widgets and save the settings. However, for non-column based layouts, you can only access the list of dashboard widgets. You need to implement widget drag and drop, and its persistence in the customized code.

Configuring a column-based layout renderer

When you add widgets to the dashboard, the dashboard framework groups the widgets with respect to their columns, and sends them to the renderer jsp as a map. The layout renderer jsp can access this map from the request using attribute name `customColumns`. The structure of the `customColumns` map is `Map<Integer>, List<DashboardWidget>`. In the layout renderer jsp, the custom column's map can be iterated to get the widgets arranged in columns. If the type of layout is `other`, that is, if the layout type is not `column`, then the map contains a single entry with a list of all the dashboard widgets.

A sample implementation of layout renderer is bundled with Business Process Portal installation. A sample four-column layout file `FourColumnsLayout.jsp` is available in the `OEBPS_HOME/webapps/deploy/sgm/bpmportal/layout/ext` folder.

To customize the sample layout renderer for ‘N’ number of columns, perform the following steps.

1. Make a copy of
   `OEBPS_HOME/webapps/deploy/sgm/bpmportal/layout/ext/FourColumnsLayout.jsp` file in the same folder and give it a different name.
2. Open the copied file in a text editor, and search for following line in the file.
   ```javascript
   var config = {
      panelItems : customPanelItems,
   };
   ```
3. Change the `colData` value to specify the width in percentage for each column. In this example, all the columns are defined for 25% of the total width.
4. Open the `dashboard-config.xml` file in a text editor, and edit it to include the new layout defined as in Step 1.
   In Business Process Portal, click ‘Create Dashboard’ or ‘Edit Dashboard’. You can see the newly added layout in the Layouts drop-down list.

Configuring a non-column-based layout renderer

You can edit the renderer JSP to place the widgets in customized layout.
To create a non-column-based dashboard layout in the renderer jsp, perform the following steps.

1. Create a jsp file containing the widget rendering logic as per your requirements in the OEBPS_HOME\webapps\deploy\sbm\bpmportal\layout\ext \ folder. Typically, it should contain the following code snippet.

```html
<%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" %>
<%@ page import="com.savvion.sbm.bpmportal.domain.DashboardWidget" %>
<%@ page import="com.savvion.sbm.bpmportal.domain.Dashboard" %>
<%@ taglib uri="http://java.sun.com/jstl/fmt" prefix="fmt" %>
<%@ taglib uri="http://java.sun.com/jstl/core" prefix="c" %>
<%@ taglib uri="/bpmportal/tld/bpmportal.tld" prefix="sbm" %>
<c:set var="customColumns" value="${customColumns}"/>
... ...

In this example, the customColumns variable contains a list of dashboard widgets that you can arrange in the custom layout.

**Note:** You need to take care of drag and drop feature and/or re-ordering of dashboard widgets for your custom dashboard layout.

2. Set the layout type to other in the dashboard-config.xml file.

The dashboard framework does not group the dashboard widgets into columns. Business Process Portal lists all dashboard widgets in the layout renderer JSP.

---

**Adding layout renderer to the dashboard configuration**

After creating a new layout renderer, you can edit the dashboard configuration file and add the new layout, with the details as described in Configuring dashboard layout on page 74. In Business Process Portal, click 'Create Dashboard' or 'Edit Dashboard'. You can see the newly added layout in the Layouts drop-down list.
Developing custom Managed Adapters

This chapter provides an overview of the interaction between components of the Managed Adapter configuration and mapping framework, and presents the The Demo Adapter on page 89, a tutorial which will guide you through the development cycle of a simple managed adapter as well as the development of the optional custom adapter configurator GUI. The tutorial also discusses the integration of the custom adapter with Progress Developer Studio for OpenEdge and covers various issues that may come up when you develop your own managed adapters.

For details, see the following topics:

- Framework operation
- Interaction of the framework components
- Adapter logging
- File formats
- Supported data types
- APIs and sample code
- Sample code of a Managed Adapter
- The Demo Adapter
- Defining the Adapter
- Defining configuration files
- Writing the Adapter classes
- Compiling and packaging the Adapter classes
- Testing and using the Adapter
Chapter 10: Developing custom Managed Adapters

- Writing the Adapter Configurator
- Handling complex configuration data
- Registering the Adapter Configurator
- Creating a custom Managed Adapter as Eclipse plugin
- Adding a library to your project
- Packaging and testing the Adapter Configurator
- Testing the new adapter configurator

Framework operation

The following sections explain the functioning of the adapter configuration and mapping framework, and describe how to build custom adapters.

Interaction of the framework components

The Figure 9 on page 81 illustrates the interactions between the various components of the framework.
The interaction of the framework components is described below:

1. **Adapter Configurator** is used to build the adapter configuration. The Adapter Configurator tool configures the adapter to perform a particular operation such as querying a specific database table, or communicating with a particular information system. The configuration also includes setting any adapter-specific configuration parameters such as temporary directories, or usernames and passwords.

2. The Adapter Configurator is an adapter-specific GUI component that is used to build the adapter configuration. Usually the Adapter Configurator GUI is started from Progress Developer Studio for OpenEdge, but it can also be started as a standalone program. The Adapter Configurator reads the adapter definition file and displays the appropriate UI.
3. Adapter Configurator stores the configuration in the file config.xml. Refer to config.xml on page 83 for description of the file.

4. Both the files, mapping.prop and config.xml are stored in the preconfigured adapter instance directory and are not yet associated to a specific BP Server Application or a Web application (formerly a BPM WebFlow process).

5. **Map Configurator.** When a Managed Adapter workstep is created, you need to define the mapping between dataslots and adapter inputs/outputs. The Map Configurator is a GUI component that allows you to define the input and output maps. It is a pluggable component, indicating that you may develop your own map configurator to handle complex or adapter-specific mapping. The Map Configurator GUI is started by Progress Developer Studio for OpenEdge.

6. **Map Deployer.** When a Business Process Server process is ready to be published, the maps and adapter definition also need to be published (or deployed). The deployment of maps and adapter definitions indicates that the data stored so far in various files is processed and transferred to a database repository, where it will be available at the process execution time. This operation is performed by the Map Deployer, which is started either automatically by Progress Developer Studio for OpenEdge or Business Process Portal, or by using the MapDeployer command-line tool.

   **Note:** If you re-install OpenEdge Business Process Server, or use the setupOEBPS utility, the deployed maps are no longer available in the repository, and you must use MapDeployer to redeploy the mapping.

7. **Run-time Mapper.** Once the maps are deployed in the database repository, the Managed Adapter can be used. When a Managed Adapter workstep is executed, BP Server or BPM WebFlow communicates to the Managed Adapter through the Run-time Mapper. The Run-time Mapper is a component that interprets the input and output maps stored in the database repository, converts the dataslots to a form acceptable to the adapter, and after the adapter execution is completed, transforms the adapter output back into a set of dataslots.

8. The Run-time Mapper is a pluggable component, indicating that you may develop your own mapper to handle complex or adapter-specific data transformations, in case the default mapper is not adequate.

## Adapter logging

Adapter logging is controlled by the log-level parameter in the mapping.prop file. The log-level is a non-negative integer, and a higher log level indicates more detailed logging. The default log level for most Managed Adapters is 10.

A separate logfile is created for each workstep that is performed by a Managed Adapter. The location of these logfiles is shown in the following table.

### Table 13: Logfile location for applications

<table>
<thead>
<tr>
<th>Application</th>
<th>Logfile location</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP Server</td>
<td>OEBPS_HOME\ebmsapps\AppName\log\WSName.log</td>
</tr>
</tbody>
</table>
Where,

- **AppName** is the BP Server Application or Web application (formerly a BPM WebFlow process) name.
- **PrivateWebAppName** is the name of the private webapp, if the adapter is used in a private webapp.
- **WSName** is the name of the Adapter workstep.

### File formats

The following sections describe the format of the files used by the Adapter Configuration and Mapping Framework and provide examples of file formats. For information on file location and usage, refer to Interaction of the framework components on page 80.

#### mapping.prop

This is the adapter definition file, containing information about the Adapter Configurator, mapping data and log level.

**Format**

```
adapter-configurator=<adapter configurator class name [optional]>
map-configurator=<map configurator class name [optional]>
runtime-mapper=<runtime mapper class name>
adapter-class=<adapter class name>
AdapterCategory=<adapter category [optional]>
AdapterName=<adapter name [optional]>
log-level=<log level [optional]>
icon=<adapter icon file name [optional]>
adaplet=<true or false [optional]>
inline-enabled=<true or false [optional]>
```

```
adapter-configurator=com.savvion.sbm.adapters.email.EmailAdapterConfigurator
map-configurator=com.savvion.sbm.adapters.framework.GroupMapConfigurator
runtime-mapper=com.savvion.sbm.adapters.framework.GroupMapper
AdapterCategory=EmailAdapters
AdapterName=GenericEmailAdapter
log-level=10
icon=mail.png
adaplet=false
```

#### config.xml

This file contains the adapter configuration. The configuration required by the default MapConfigurator is stored in XML, but a custom implementation of a MapConfigurator may use other custom format.
This file is stored in the `OEBPS_HOME\ebmsapps\<Application_Name>\maps\<WS_Name>` directory, where `WS_Name` is the name of the workstep.

The XML file contents are:

```xml
<configform synchronous="true|false">
  <block title="title" description="description" hidden="true|false">
    <param name="name" access="I|O|IO|HIDDEN|PASSWORD" description="description" type="java_type" value="default_value"/>
    ...
  </block>
  ...
</configform>
```

The schema is given below:

```xml
<?xml version="1.0" encoding="utf-16"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="configform">
    <xs:complexType>
      <xs:attribute name="synchronous" type="xs:string" use="optional"/>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="block">
          <xs:complexType>
            <xs:attribute name="title" type="xs:string" use="optional"/>
            <xs:sequence>
              <xs:element maxOccurs="unbounded" name="param">
                <xs:complexType>
                  <xs:attribute name="name" type="xs:string" use="required"/>
                  <xs:attribute name="access" type="xs:string" use="required"/>
                  <xs:attribute name="description" type="xs:string" use="optional"/>
                  <xs:attribute name="type" type="xs:string" use="optional"/>
                  <xs:attribute name="value" type="xs:string" use="optional"/>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

The interpretation of the schema elements is shown in the following table.

**Table 14: Default ConfigForm format**

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>`&lt;configform synchronous=&quot;true</td>
<td>false&quot;&gt; ...&lt;/configform&gt;`</td>
</tr>
</tbody>
</table>
<block title="title" description="description" hidden="true|false">

[required] Defines a block of adapter parameters. One or more blocks can be defined in a ConfigForm. Grouping of the parameters in a block is useful for several purposes:

• Visual grouping of the parameters later in the MapConfigurator GUI.
• Defining a group of parameters as "hidden", not visible from the MapConfigurator GUI.

The "title" [required] attribute is used solely for visual purposes, to group the contained properties in a labeled block. The "description" [optional] attribute is not used in the default GUI, but may be used in extended GUI versions. The "hidden" [optional] attribute defines the visibility of all the parameters in the block. When this parameter is not specified, the value is taken as "true".

<param name="name" access="I|O|O|IO|HIDDEN|PASSWORD" description="description" type="java_type" value="default_value"/>

[required] Each block contains one or more adapter parameters. The parameters represent adapter inputs, outputs and configuration values. The "name" [required] attribute defines parameter's name. The "access" [required] attribute defines how the parameter is to be used:

• I – adapter input
• O – adapter output
• IO – input/output
• HIDDEN – hidden parameter (for example a configuration constant)
• PASSWORD - the same as an IO field, except that it is displayed as a password field.

The "description" [optional] attribute can be used by the MapConfigurator GUI to display short description or help message about the parameter. The "type" [optional] attribute is used by the mapper to specify the Java data type of the parameter. When this parameter is not specified, the type is assumed as "java.lang.String". The "value" [optional] attribute gives the default value of the parameter.

**input.map and output.map**

These two files are used to store the input and output maps for an adapter. The files are located in the `<Process_Directory>\maps\<WS_Name>` directory.

The information in this section is given as a reference only. Neither the Business Process Server users nor adapter developers are expected to directly manipulate the mapping files.

Adapters introduced with OpenEdge Business Process Server store the input and output map files in XML. Typically, the new tabbed map configurator (com.savvion.sbm.adapters.framework.TabbedMapConfigurator) generates XML files, while the simpler map configurators (for example, com.savvion.sbm.adapters.framework.TwoWayMapConfigurator) generate property files. The property format is now deprecated, and although it is still supported, maps in property file format will be automatically converted to XML.

The following example of the ConfigForm generates the input/output maps.
Chapter 10: Developing custom Managed Adapters

Config Form

```xml
<configform>
  <block title="Adapter Data" hidden="false">
    <param name="Price" access="IO" type="java.lang.Object" value=""/>
    <param name="VAT" access="I" type="java.lang.Object" value=""/>
  </block>
  <block title="Configuration" hidden="false">
    <param name="WORKBOOK" access="IO" description="Workbook" type="java.lang.String" value="M:\conf\VAT.xls"/>
    <param name="WORKSHEET" access="IO" description="Default worksheet" type="java.lang.String" value="Sheet1"/>
    <param name="INPUTS" access="HIDDEN" description="Input cells list" type="java.lang.String" value="Price=A1,VAT=B1"/>
    <param name="OUTPUTS" access="HIDDEN" description="Output cells list" type="java.lang.String" value="Price=C1"/>
  </block>
</configform>
```

Input Mapping

```
Price=java.lang.Object:@AskingPrice
VAT=java.lang.Object:0.08
WORKBOOK=java.lang.String:M:\conf\VAT.xls
WORKSHEET=java.lang.String:Sheet1
INPUTS=java.lang.String:Price=A1,VAT=B1
OUTPUTS=java.lang.String:Price=C1
```

Output Mapping from the Adapter to Progress Developer Studio for OpenEdge

```
UpdatedPrice=java.lang.Double:@Price
```

XML Map format

As mentioned before, the Managed Adapters in the current release use XML files to store the mapping data. The file names and locations are the same as prior releases, but the XML content is in the following format:

- **Input Map**

  ```xml
  <map version="1.1">
    <param name="AdapterParameterName" type="java_type" value="value"/>
    ...
  </map>
  ```

- **Output Map**

  ```xml
  <map version="1.1">
    <param name="DataslotName" type="java_type" value="value"/>
    ...
  </map>
  ```
Supported data types

Adapter Configuration and Mapping Framework supports two-way conversion between various data types and boolean, as listed in the following table.

Table 15: Supported types for conversion

<table>
<thead>
<tr>
<th>Source type</th>
<th>Target type</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.lang.String</td>
<td>S SF SF SF SF SF SF SF SF S</td>
</tr>
<tr>
<td>java.lang.Long</td>
<td>SC S SC SC SC SC SC SC S</td>
</tr>
<tr>
<td>java.lang.Integer</td>
<td>SC SC S SC SC SC SC SC S</td>
</tr>
<tr>
<td>java.lang.Double</td>
<td>SC SC SC S SC SC SC SC S</td>
</tr>
<tr>
<td>java.lang.Float</td>
<td>SC SC SC SC S SC SC SC S</td>
</tr>
<tr>
<td>java.math.BigDecimal</td>
<td>SC SC SC SC SC S SC SC S</td>
</tr>
<tr>
<td>java.sql.Timestamp</td>
<td>SC SC SC SC SC S</td>
</tr>
<tr>
<td>java.lang.Boolean</td>
<td>SC SC SC SC S S</td>
</tr>
<tr>
<td>java.lang.Object</td>
<td>SC SF SF SF SF SF SF SF S</td>
</tr>
</tbody>
</table>

- **S**: supported, no conversion performed
- **SF**: supported, if the source is in proper format
- **SC**: supported, conversion performed

When a value is converted from one data type to another, following rules apply:

- **Converting numeric values.** Care must be exercised while converting between different numeric types. The transformation may involve loss of precision, or the value may be truncated. For example, if an adapter output is defined as java.lang.Double, you may still map it to an integer data type, but in this case the value will be rounded off.

  Note: All numeric types except "BigDecimal" are internally stored as "Double" values, which can have an effect when trying to convert values outside the range of the target type.

- **Converting between boolean and numeric types.** When a number is converted to Boolean, any non-negative value will be converted to "true"; negative or zero value will be converted to "false". The reverse conversion, from boolean to numeric, maps "true" to "1" and "false" to "0".

- **Mapping to "java.lang.Object".** If any value is mapped to "java.lang.Object", no transformation will be performed.
• **Mapping <null> values.** When a mapped input dataslot contains a null value, the corresponding input will not be defined in the adapter inputs hashtable. When a Managed Adapter returns a null value for a mapped output, the value of the corresponding dataslot will be set to the dataslot default value by BP Server.

## APIs and sample code

This section describes the various supported APIs and sample code for them.

### Supported APIs and sample code

**• com.savvion.sbm.adapters.framework.ManagedAdapter**

```java
public abstract class ManagedAdapter {
    protected Object input;
    protected Object output;
    public final static int LOG_ERROR = 0;
    public final static int LOG_WARNING = 1;
    public final static int LOG_MESSAGE = 2;
    public final static int LOG_TRACE = 3;
    public void log(int logLevel, String message);
}
```

**• com.savvion.sbm.adapters.framework.AdapterConfigurator**

```java
public interface AdapterConfigurator {
    public void configForm(Hashtable context) throws Exception;
}
```

**• com.savvion.sbm.adapters.framework.MapConfigurator**

```java
public interface MapConfigurator {
    public void buildMap(Hashtable context, Hashtable inputDataslots, Hashtable outputDataslots) throws Exception;
    public void buildMap(Hashtable context, Hashtable inputDataslots, Hashtable outputDataslots, Component parent) throws Exception;
    public void init(Hashtable context) throws Exception;
    public void loadConfig() throws Exception;
    public void loadMaps() throws Exception;
    public void prepare(Hashtable inputDataslots, Hashtable outputDataslots) throws Exception;
    public void prepareAndShowGUI(Component parent) throws Exception;
    public void save(Hashtable context) throws Exception;
    public Collection getInputDataslots() throws Exception;
    public Collection getOutputDataslots() throws Exception;
}
```

**• com.savvion.sbm.adapters.framework.Mapper**

```java
public interface Mapper {
    public void setMapinfo(Hashtable data) throws Exception;
    public Object mapIn(Hashtable data) throws Exception;
    public Hashtable mapOut(Object data) throws Exception;
    public String getInputSource(String input) throws Exception;
    public String getOutputDestination(String output) throws Exception;
    public String getOutputDestinationType(String output) throws Exception;
}
```
Sample code of a Managed Adapter

This code assumes that the Managed Adapter is designed to work with the default Run-time Mapper.

Sample code of Managed Adapter

```java
package com.savvion.sbm.adapters.sample;
import com.savvion.sbm.adapters.framework.*;
import java.util.*;
public class SampleAdapter extends ManagedAdapter {
  public void execute() throws Exception {
    // Get the input data
    Hashtable inputs = (Hashtable)input;
    // Log a message
    log(LOG_TRACE,"Got input data...");
    // Prepare the output hashtable
    Hashtable outputHashtable = new Hashtable();
    ... do some processing and populate outputHashtable ...
    // Set the adapter output
    output = outputHashtable;
  }
}
```

The Demo Adapter

In this tutorial we will develop a simple adapter that, given two numbers, can perform one of the four basic arithmetic operations – addition, subtraction, multiplication, or division – and return the result. Optionally, the adapter can round the result to the nearest integer number. Which arithmetic operation will be used, and whether the result should be rounded or not, are parameters that can be configured on a per-workstep basis.

The following diagram illustrates the adapter functionality:

**Figure 10: The Demo Adapter**

As shown above, “X” and “Y” are the two inputs, that are usually mapped to Business Process Server dataslots. The “OPERATION” and “ROUNDING” are configuration parameters. Together, all inputs determine the value of the “RESULT” output parameter.
Defining the Adapter

Before any managed adapter can be used in OpenEdge Business Process Server, the adapter should be described using the adapter metadata file. All metadata files are located in the .com.savvion.studio/adapters directory located in the Progress Developer Studio for OpenEdge workspace folder. There, the managed adapters are organized in groups, each group being a set of adapters with similar functionality.

In order to create a separate group, create a directory named DemoAdapters under <workspace>/.com.savvion.studio/adapters. DemoAdapters will be the name of the new adapter group.

Next, we should create the adapter directory for our new adapter. Under the <workspace>/.com.savvion.studio/adapters/DemoAdapters directory, create a new subdirectory named MathAdapter. This will be the adapter name. The adapter metadata files are stored in this directory.

The first file that we have to create is mapping.prop. This file defines all the basic properties of the adapter, such as run-time class, type of mapping, and log-level.

In the <workspace>/.com.savvion.studio/adapters/DemoAdapters/MathAdapter directory, create a file named mapping.prop with the content shown below (alternatively, you can copy an existing mapping.prop file from some other adapter in the managedadapters directory, and just edit the property values):

```
adapter-class=demo.MathAdapter
adapter-configurator=
map-configurator=com.savvion.sbm.adapters.framework.GroupMapConfigurator
runtime-mapper=com.savvion.sbm.adapters.framework.GroupMapper
AdapterCategory=DemoAdapters
AdapterName=MathAdapter
log-level=10
```

The meaning of each property is given in the table below:

**Table 16: Information on mapping.prop**

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>adapter-class</td>
<td>The fully qualified name of the adapter class. This is the class that will be invoked by OpenEdge Business Process Server at run time.</td>
</tr>
<tr>
<td>adapter-configurator</td>
<td>The fully qualified name of the adapter configurator GUI class (if the adapter has a configurator). This class is invoked by Progress Developer Studio for OpenEdge, when the adapter is configured. In our case, the adapter-configurator parameter is empty, meaning that, at least at this stage, we will not have an adapter configurator GUI, but will use a static configuration file.</td>
</tr>
<tr>
<td>map-configurator</td>
<td>The fully qualified name of the map configurator GUI class. This class is provided by OpenEdge Business Process Server. We recommend always using the GroupMapConfigurator.</td>
</tr>
</tbody>
</table>
Defining configuration files

Most managed adapters have a custom adapter configurator GUI. This GUI is invoked by Progress Developer Studio for OpenEdge when the adapter is used in a workstep, and allows the user to specify various configuration parameters.

Alternatively, if the adapter is simple, it may not need the customized configurator GUI. A static configuration file can be used instead. In this tutorial, we will demonstrate both approaches, but for simplicity, let’s start with defining a static configuration file first.

The file will define all the adapter inputs, outputs, and configuration parameters, and will also describe how this data must be presented in the map configurator GUI (this GUI is provided by OpenEdge Business Process Server).

Following is the list of parameters for the Math adapter (see Figure 12 on page 97):

<table>
<thead>
<tr>
<th>Parameter(s)</th>
<th>Access</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X, Y</td>
<td>input</td>
<td>INTEGER</td>
<td>The two input values.</td>
</tr>
<tr>
<td>Operation</td>
<td>configuration (hidden input)</td>
<td>CHARACTER</td>
<td>The arithmetical operation to be performed (one of add, sub, mul, or div)</td>
</tr>
<tr>
<td>Rounding</td>
<td>configuration (hidden input)</td>
<td>CHARACTER</td>
<td>Set to “true” if the result should be rounded, “false” otherwise.</td>
</tr>
<tr>
<td>Result</td>
<td>output</td>
<td>INTEGER</td>
<td>The result of the operation performed on the input numbers.</td>
</tr>
</tbody>
</table>

Defining configuration files

Most managed adapters have a custom adapter configurator GUI. This GUI is invoked by Progress Developer Studio for OpenEdge when the adapter is used in a workstep, and allows the user to specify various configuration parameters.

Alternatively, if the adapter is simple, it may not need the customized configurator GUI. A static configuration file can be used instead. In this tutorial, we will demonstrate both approaches, but for simplicity, let’s start with defining a static configuration file first.

The file will define all the adapter inputs, outputs, and configuration parameters, and will also describe how this data must be presented in the map configurator GUI (this GUI is provided by OpenEdge Business Process Server).

Following is the list of parameters for the Math adapter (see Figure 12 on page 97):

<table>
<thead>
<tr>
<th>Parameter(s)</th>
<th>Access</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>X, Y</td>
<td>input</td>
<td>INTEGER</td>
<td>The two input values.</td>
</tr>
<tr>
<td>Operation</td>
<td>configuration (hidden input)</td>
<td>CHARACTER</td>
<td>The arithmetical operation to be performed (one of add, sub, mul, or div)</td>
</tr>
<tr>
<td>Rounding</td>
<td>configuration (hidden input)</td>
<td>CHARACTER</td>
<td>Set to “true” if the result should be rounded, “false” otherwise.</td>
</tr>
<tr>
<td>Result</td>
<td>output</td>
<td>INTEGER</td>
<td>The result of the operation performed on the input numbers.</td>
</tr>
</tbody>
</table>
As we already defined in the `mapping.prop` file, we will use the GroupMapConfigurator GUI. It allows, when the adapter inputs and outputs are mapped to dataslots, parameters to be grouped under several tabs for better presentation and logical structure. In our case it is convenient to have the "X" and "Y" input parameters in one tab, and the "RESULT" parameter in another tab. The "OPERATION" and "ROUNDING" parameters must also be given to the map configurator and since we do not want them to be visible when dataslot mapping is done, we can put them in a separate "hidden" tab.

All this information is specified in the configuration file. In the `<workspace>/.com.savvion.studio/adapters/DemoAdapters/MathAdapter` directory, create a file named `config.xml` with the following content:

```
<configform>
  <block title="Config" description="Config" hidden="true">
    <param name="%OPERATION%" access="HIDDEN" description="OPERATION" type="java.lang.String" value="add"/>
    <param name="ROUNDING" access="HIDDEN" description="ROUNDING" type="java.lang.String" value="false"/>
  </block>
  <block title="Inputs" description="Inputs">
    <param name="X" access="I" description="X" type="java.lang.Double" mandatory="true"/>
    <param name="Y" access="I" description="Y" type="java.lang.Double" mandatory="true"/>
  </block>
  <block title="Outputs" description="Outputs">
    <param name="RESULT" access="O" description="RESULT" type="java.lang.Double"/>
  </block>
</configform>
```

Here, enclosed by a pair of "<configform> … </configform>" tags are three "blocks". Each block will be presented as a separate tab by the GroupMapConfigurator GUI and will contain a logical group of input or output parameters. The exception is the block containing the configuration parameters "OPERATION" and "ROUNDING", which we do not want to show as a tab in the map configurator.

For each of the three blocks – "Inputs", "Outputs", and "Config" – we must provide title and description. The title is used to uniquely identify the block, and the description is a short label, that will be displayed on the corresponding tab.

Note that the "Config" block has one extra attribute – hidden="true". When set to "true", this attribute tells the map configurator not to display the tab, or the parameters in it.

The parameters are listed in each block. Each parameter is defined by a name (used for identification), access (attribute specifying if the parameter is input, output or a hidden configuration parameter), description (a short label displayed by the GUI for this parameter), data type (fully qualified java class name, one of the supported types listed in Supported data types on page 87) and optionally, a default value.

Although we have not yet written the adapter itself, the following two figures provide a preview of how the map configurator GUI will interpret and represent this `config.xml` file (that is, when the completed adapter is used):
Writing the Adapter classes

The next thing to do is to write the adapter run-time classes. Let's first define the constants that will be used in our adapter, in the demo.MathAdapterConstants interface:

MathAdapterConstants.java

```java
package demo;
public interface MathAdapterConstants {
    public final static String OPERATION_ADD = "add";
    public final static String OPERATION_SUB = "sub";
    public final static String OPERATION_MUL = "mul";
    public final static String OPERATION_DIV = "div";
    public final static String PARAM_OPERATION = "OPERATION";
    public final static String PARAM_ROUNDING = "ROUNDING";
    public final static String PARAM_X = "X";
    public final static String PARAM_Y = "Y";
    public final static String PARAM_RESULT = "RESULT";
}
```

Here, we defined strings for the various operations as well as the names of all the input, output and configuration parameters.

The actual adapter class is shown in the following code sample:
Note: For the sake of simplicity, in the above source we have omitted many fault checks and safeguards (for example, checking for illegal division by zero).

Each managed adapter must extend the `com.savvion.sbm.adapters.framework.ManagedAdapter` class (line 6).

A custom managed adapter is expected to overwrite the "execute()" method (lines 10-37). Due to the backward compatibility requirements, the "execute()" method does not have parameters and does not return a value. Instead, a pair of Objects, "input" and "output", that are defined in the ManagedAdapter base class, are used to move data in and out of the adapter. Although "input" and "output" can be of any Serializable java type, when using the default Business Process Server mappers (as in this case), the content of the "input" object is a java.util.Hashtable and the "output" object is expected to be set to another Hashtable from the adapter. In order to avoid type casting in the adapter code, we mapped those two objects to two Hashtables – "inputs" and "outputs" (line 11 and lines 34-36).
Note: The data type of the objects, contained in the input Hashtable will match the types specified in the "config.xml" file. For example, the "X" parameter will always come to the adapter as "java.lang.Double", because this is how it was defined in "config.xml" (see Writing the Adapter Configurator on page 96). When the data type of a dataslot does not match the required data type of an output, the adapter framework will take care of the necessary conversions. List of the supported data type conversions is given in Supported data types on page 87.

The "inputs" Hashtable will contain all adapter inputs and configuration parameters. On lines 13-14 we extract from the "inputs" Hashtable the "X" and "Y" input parameters. The operation name (a configuration parameter) is also supplied in the "inputs" Hashtable (line 17).

Lines 18-29 use the "OPERATION" parameter and the "X" and "Y" inputs to calculate the result. In case the operation name is invalid (it may happen, if the adapter is not properly configured), an error message is logged in the adapter's own log file (located in the OEBPS_HOME/ebmsapps/<application>/logs directory on the server) and an exception is thrown. Throwing an exception allows the workstep to become suspended and also logs an error message in the Progress Developer Studio for OpenEdge log file.

Note: The "log()" method, provided by the ManagedAdapter base class, accepts two parameters — an integer log-level, and the message text string. The log-level can be any non-negative integer; a higher number indicates a lower-priority message. Messages will be logged in the adapter file only if the log-level of the adapter, specified in the mapping.prop file, is higher than or equal to the log-level of the message.

On lines 31-32, we check the value of the "ROUNDING" configuration parameter. If this parameter is set to "true", the result is rounded to the nearest integer.

Finally, on lines 34-36, we store the result in the "RESULT" output parameter. Again, note the way the Hashtable is returned through the "output" Object.

Compiling and packaging the Adapter classes

When compiling your custom adapter, the only OpenEdge Business Process Server-related jar you need in the classpath is the oebpsadapterframework.jar, located in the OEBPS_HOME/lib directory on the server. This is the jar that contains the ManagedAdapter base class and the other files from the com.savvion.sbm.adapters.framework package.

The out-of-the-box adapters supplied with OpenEdge Business Process Server come packaged as separate jar-files. The jar-files contain all adapter classes, including GUI components. You are not required to follow this procedure. For example, you may decide to store your adapter's classes unpackaged in the OEBPS_HOME/ebmsapps directory. In any case, make sure that your adapter classes are available to the server.
Testing and using the Adapter

After compiling and making the classes available to OpenEdge Business Process Server, you can test the adapter. Create a BP Server application that uses the adapter. In the Properties dialog window for the Adapter workstep, select the Fields tab and click the Custom dataslot mapping... button. Define the input and output dataslot mapping, then save, publish, and run the application to confirm that you are getting the expected results.

If you try to start the adapter configurator, an error message should appear, telling you that the adapter does not have a pluggable configurator. This is expected behavior, because we do not have a custom configurator GUI for the Math adapter.

Since we do not have a custom configurator GUI, the only way to change the adapter configuration (the "OPERATION" and "ROUNDING" parameters) is to edit the config.xml file directly.

Optional Approaches

If you do not want to define custom adapter configurator, there are two more ways to avoid editing the config.xml file:

The first option is to change the "access" attribute of the configuration parameters in the config.xml file from "HIDDEN" to "I" (see the sample config.xml on page 83). Make the block where these parameters are located visible by changing the "hidden" attribute to false. This will enable the Map Configurator to show the configuration parameters in a separate tab. The users of your adapter (the designers of OpenEdge Business Process Server applications) will then be able to see and change the default values of these parameters in the map configurator GUI provided by the adapter framework.

Another option is to have several preconfigured instances of your adapter. For example, you can have four separate configurations of the Math adapter, each performing one of the basic arithmetic operations. To do this, make four copies of the MathAdapter directory under the OEBPS_HOME/managedadapters/Demo directory (for example, "MathAdd", "MathSub", "MathMul", and "MathDiv"). Modify the mapping.prop file in each directory to match the adapter name (the name will be "MathAdd" in the "MathAdd" directory, and so on). Modify the config.xml files for each adapter instance in the way you want. When a user (OpenEdge Business Process Server application developer) chooses one of these preconfigured adapters, the appropriate configuration file will be used in the workstep.

When testing the adapter, look for errors and log-messages in the adapter log file, OEBPS_HOME/ebmsapps/<application>/logs/<workstep>.log, on the server. You can change the log-level in mapping.prop file by opening the Workstep Properties dialog and clicking the "Advanced properties..." button in the Configuration tab, and republishing the OpenEdge Business Process Server application. Log-levels 11 and 12 will dump a lot of debug information from the Managed Adapter framework, including a full configuration dump and a complete list of what is passed to and retrieved from the adapter.

Writing the Adapter Configurator

If an adapter is going to be used often, it is not convenient each time to manually edit the "config.xml" file in order to change the adapter configuration. Developing a custom adapter configurator GUI solves this problem and also allows more complex configurations to be used.
A custom adapter configurator for the Math adapter should allow the user to select the operation and to turn on and off the rounding of the result. A simple GUI that does this is shown in the following figure:

**Figure 12: MathAdapter Configurator GUI**

![MathAdapter Configurator GUI](image)

The task of an adapter configurator is to generate the config.xml file according to the user's inputs. A managed adapter configurator is a class that extends the "com.savvion.sbm.adapters.framework.AdapterConfiguratorBase" class and overwrites the following methods:

- **public void init(JPanel panel)**. This method creates the GUI. An empty JPanel is passed by the framework and the custom adapter configurator is expected to populate it with GUI elements. Those must not include the **OK** and **Cancel** button, which are placed automatically on every adapter configurator dialog. In the case of the MathAdapter configurator, the init() method should place in the JPanel the operation and rounding labels and check boxes, as shown in the above figure.

- **public void setConfig(AdapterConfig ac)**. This method is called by the adapter framework if there is an existing configuration (for example, if the adapter configuration for an already existing Business Process Server process is edited). The "ac" parameter is a "com.savvion.sbm.adapter.framework.AdapterConfig" object – a wrapper around the config.xml format. It allows you, as an adapter developer, to avoid parsing XML format and shields you for any future changes in the XML DTD.

- **public AdapterConfig getConfig()**. This method is called by the adapter framework when the user (OpenEdge Business Process Server process developer) presses the **OK** button on the adapter configurator. The adapter configurator is expected to return a "com.savvion.sbm.adapters.framework.AdapterConfig" object, describing the config.xml file. Your code does not have to worry when, how, and where this information is stored. See the comments regarding the AdapterConfig object in the description of the setConfig() method above.

Following is a listing of the MathAdapter configurator implementing these three methods:

```java
01 package demo;
02 import com.savvion.sbm.adapters.framework.AdapterConfig;
03 import com.savvion.sbm.adapters.framework.AdapterConfiguratorBase;
04 import java.awt.GridBagConstraints;
05 import java.awt.GridBagLayout;
06 import java.awt.Insets;
07 import javax.swing.JComboBox;
08 import javax.swing.JLabel;
09 import javax.swing.JPanel;
10 import javax.swing.JTextField;

14 public class MathAdapterConfigurator extends AdapterConfiguratorBase
15     implements MathAdapterConstants
16 {
17     private JLabel OPERATION_LABEL;
18     private JLabel ROUNING_LABEL;
19     private JComboBox OPERATION_COMBO_BOX;
20     private JComboBox ROUNING_COMBO_BOX;
```
public void init(JPanel panel) throws Exception {
    setTitle("MathAdapter Config");
    OPERATION_LABEL = new JLabel("Operation");
    ROUNING_LABEL = new JLabel("Rounding");
    OPERATION_COMBO_BOX = new JComboBox{
        new String[] {
            OPERATION_ADD,
            OPERATION_SUB,
            OPERATION_MUL,
            OPERATION_DIV
        });
    ROUNING_COMBO_BOX = new JComboBox{
        new String[]{"false", "true"}
    };
    panel.setLayout(new GridBagLayout());
    GridBagConstraints cs = new GridBagConstraints();
    cs.insets = new Insets(5,5,5,5);
    cs.fill = cs.HORIZONTAL;
    cs.weightx = 0.0;
    cs.gridx = cs.gridy = 0;
    panel.add(OPERATION_LABEL, cs);
    cs.gridy = 1;
    panel.add(ROUNDING_LABEL, cs);
    cs.weightx = 1.0;
    cs.gridx = 1; cs.gridy = 0;
    panel.add(OPERATION_COMBO_BOX, cs);
    cs.gridy = 1;
    panel.add(ROUNDING_COMBO_BOX, cs);
}
public void setConfig(AdapterConfig ac) throws Exception {
    OPERATION_COMBO_BOX.setSelectedIndex(0);
    ROUNDING_COMBO_BOX.setSelectedIndex(0);
    if (ac!=null) {
        String operation = ac.getParameterValue(PARAM_OPERATION);
        if (operation!=null)
            OPERATION_COMBO_BOX.setSelectedItem(operation);
        String rounding = ac.getParameterValue(PARAM_ROUNDING);
        if (rounding!=null)
            ROUNDING_COMBO_BOX.setSelectedItem(rounding);
    }
}
public AdapterConfig getConfig() throws Exception {
    AdapterConfig ac = new AdapterConfig();
    String operation = (String)OPERATION_COMBO_BOX.getSelectedItem();
    String rounding = (String)ROUNDING_COMBO_BOX.getSelectedItem();
    ac.addBlock("Config", "Config", true);
    ac.addParameter("Config", PARAM_OPERATION,
    AdapterConfig.ACCESS_HIDDEN,
    "java.lang.String",
    operation,
    "true");
    ac.addParameter("Config", PARAM_ROUNDING,
    AdapterConfig.ACCESS_HIDDEN,
    "java.lang.String",
    rounding,
    "true");
}

In the init() method, lines 25-38 initialize the GUI elements. Those are then placed in the JPanel at lines 40-53. Again, note that this does not include the OK and Cancel buttons – OK and Cancel are placed under the JPanel by the adapter framework.

The setConfig() method, on lines 57-58 initializes the check boxes with their default values. If the AdapterConfig parameter is not null, the check boxes are reset with the values from this existing configuration (lines 59-66).

The longest method in this class, getConfig(), generates a new AdapterConfig object, based on the user's (the OpenEdge Business Process Server process designer) entries in the configurator. A fresh AdapterConfig is created and the values for the "OPERATION" and "ROUNDING" parameters are obtained on lines 70-73.

Next, each of the blocks in the adapter configuration, and the parameters in each block, are added in turn. Refer to the section on config.xml on page 83. Note that this method defines not only the "OPERATION" and "ROUNDING" configuration parameters, but also specifies all adapter inputs and outputs – in this case, the "X", "Y" and "RETURN" parameters. The difference is that the "X", "Y", and "RETURN" parameters do not have default value, and are made visible for mapping to dataslots by the user.

Handling complex configuration data

The configuration data for the MathDemo is quite simple – just two config parameters, "OPERATION" and "ROUNDING". Some adapters, however, may require much more complex configuration that will make the approach of defining the configuration parameters one by one in the config file impractical.

A better approach in this case is to use your own custom configuration object that has its own XML representation. Your custom object must know how to generate the XML representation and how to initialize itself if given such an XML representation string. If you use a JavaBean style object to hold your configuration, there are a number of java libraries that can serialize/deserialize your object to/from XML (for example "java.beans.XMLEncoder" and "java.beans.XMLDecoder").
If you have such an object, you can store the XML string as a value of a single configuration parameter. For example, you can include the following in your adapter configurator class:

```java
ac.addParameter("ConfigBlock",
    PARAM_CONFIG,
    AdapterConfig.ACCESS_HIDDEN,
    PARAM_CONFIG,
    "java.lang.String",
    xmlConfigurationString);
```

Here, "xmlConfigurationString" contains the XML-serialized value of your custom configuration object.

When the configuration is retrieved, either by the adapter run time or by the adapter configuration opening an existing configuration in the `getConfig()` method, you can restore your original object from the XML string. The XML string can be obtained from the `AdapterConfig` object as follows:

```java
xmlConfigurationString = ac.getParameterValue(PARAM_CONFIG);
```

## Registering the Adapter Configurator

When you compile the custom adapter configurator and make it available in the OpenEdge Business Process Server classpath (see Compiling and packaging the Adapter classes on page 95), you must change the mapping.prop file to list the adapter configurator class before using it in Progress Developer Studio for OpenEdge.

Open the `mapping.prop` file from the `<workspace>/.com.savvion.studio/adapters/DemoAdapters/MathAdapter` directory, and give the configurator class name in the `adapter-configurator` property, as shown below:

```
mapping.prop

adapter-class=demo.MathAdapter
adapter-configurator=demo.MathAdapterConfigurator
map-configurator=com.savvion.sbm.adapters.framework.GroupMapConfigurator
runtime-mapper=com.savvion.sbm.adapters.framework.GroupMapper
AdapterCategory=DemoAdapters
AdapterName=MathAdapter
log-level=10
```

This will make the adapter configurator available from Progress Developer Studio for OpenEdge.

Note that in case you have an already existing application using the MathAdapter, this change will not be automatically propagated to the worksteps in this application. You will have to change the configuration (`Workstep Properties dialog > Configuration > Advanced Properties...`) and republish the affected applications.

## Creating a custom Managed Adapter as Eclipse plugin

You can efficiently manage a custom managed adapter interface by creating an adapter configurator as an Eclipse plugin in Progress Developer Studio for OpenEdge.

To create an adapter configurator as a Progress Developer Studio for OpenEdge plugin:
Creating a custom Managed Adapter as Eclipse plugin

1. From File menu in Progress Developer Studio for OpenEdge, click New > Other, opening the New wizard.
2. Expand Plug-in Development, select Plug-in Project, then click Next.
3. From the New Plug-in Project wizard, enter the project name (in this case, "DemoAdapter").
4. Proceed to create the plug-in project by completing the rest of wizard steps with default settings.
5. After creating the project, open the “plugin.xml” editor.
6. Add the "com.savvion.studio.adapters" plug-in as a dependency in the Dependency tab (as shown in Figure 13 on page 101).

**Figure 13: Adding dependency**

![Dependencies](image)

7. From the Extensions tab, click Add to define the extension point in the New Extension wizard.
Figure 14: New Extension wizard

a) Select "com.savvion.adapter.framework" extension, then click Finish.
b) From the Extension Details section of the Extensions tab, enter a unique ID ("demoAdapterPlugin") and Name ("Demo Adapter Plugin") in the respective boxes.
c) Add the plug-in configuration by editing the XML in the plugin.xml tab. Figure 15 on page 103 highlights the added <config> tag with the "configuratorClass" attribute defining the adapter configurator class name.
d) Switch to the Extensions tab to view the added extension element.

Adding a library to your project

To add the oebpsadapterframework.jar library to your project, so that the project is successfully compiled and exported as an Eclipse plugin:
1. For the plug-in project, create the \lib folder and copy the oebpsadapterframework.jar file from Studio_Home\eclipse\plugins\.com.savvion.studio.resources_7.5.2.<build_timestamp>\lib folder to this folder.

2. From the project's Properties, click Java Build Path > Libraries > Add JARs to add the \lib\oebpsadapterframework.jar file.

3. Add the oebpsadapterframework.jar file to plug-in runtime classpath in the Runtime tab of the "plugin.xml" editor for this project.
   a) From the Classpath section of the Runtime tab, click Add to add the \lib\oebpsadapterframework.jar file.

You can now create the adapter configurator class for this project by extending the com.savvion.sbm.adapters.framework.AdapterConfiguratorBase super class (as shown in Figure 17 on page 104).

Figure 17: Creating Adapter Configurator class

```
Ensure that you use the same package and class name as specified in the "configuratorClass" attribute, while editing the plugin.xml (Figure 15 on page 103). After creating, configure the adapter configurator, for instance, as following for this project:
```

```java
package com.savvion.demo;
import javax.swing.JLabel;
import javax.swing.JPanel;
import com.savvion.sbm.adapters.framework.AdapterConfiguratorBase;
public class DemoAdapterConfigurator extends AdapterConfiguratorBase {
    private static final long serialVersionUID = 1L;
```
Packaging and testing the Adapter Configurator

Before testing the adapter configurator in Progress Developer Studio for OpenEdge, you need to package it as a deployable plugin JAR.

1. From the Project Explorer view, right-click the adapter plugin project (in this case, DemoAdapter), then click Export to launch the Export wizard.
2. Expand Plug-in Development, select Deployable plug-ins and fragments, then click Next.
3. From the second page of the Export wizard, select the adapter plugin project to be exported.

Figure 18: Export wizard page 2
a) From the **Destination** tab, select the directory where you want to generate the plugin JAR (in this case, **DemoAdapter_1.0.0.jar**). You can copy the generated plugin JAR to **Studio_Home\eclipse\plugins** folder later or directly specify this folder here.

b) Click **Finish** to create the deployable plugin JAR file in the specified folder.

4. From the Project Explorer, copy the plug-in project folder (in this case, **DemoAdapters**) from **<project_name>\adapters** folder to **Studio_Home\workspace\.com.savvion.studio\adapters** folder.

---

**Testing the new adapter configurator**

**To test the new adapter configurator**

Restart Progress Developer Studio for OpenEdge and launch the **Managed Adapter Browser** tool from **Tools > Managed Adapters**.

You can view the new adapter in the existing list of managed adapters.

---

**Note:** For information regarding operations in the Managed Adapter Browser tool, see Chapter 25: "Using Managed Adapter Browser" in *OpenEdge Getting Started: Developing BPM Applications with Developer Studio*.
Localizing Business Process Server

You can localize Business Process Server components, namely Business Process Portal, BPM Designer, and Business Process Modeler, to a language of your choice. We recommend that a Senior Application Developer with expertise in Java design and development performs this localization task. This chapter details the steps to be taken to localize each Business Process Server component.

Before you start localizing Business Process Server component, ensure that your computer is configured to use the requisite language. Also make sure that you select appropriate fonts and character encoding for your browser. For example, if you want to localize to Japanese language, you must set the fonts and character encoding for the Mozilla Firefox browser as shown in the following figure.
Figure 19: Fonts and Character Encoding for Japanese in Firefox

For more details about these, refer to your operating system’s documentation.

**Note:** To ensure that Task notification emails (using a localized HTML or ETL Email template) display language characters properly, you need to set the `sbm.email.outgoing.server.charset` parameter in the `oebpsemail.properties` file and the `oebps.characterencoding` parameter in the `oebps.conf` file to value, UTF-8.

For details, see the following topics:

- Localizing Business Process Portal
- Specifying the date formats
- Translating the Properties file
- Customizing the interface labels
- Adding support in Business Process Portal
- Localizing dashboard
- Localizing dashboard widget title and description
- Localizing dashboard title and description
- Configuring dashboard widget height
Localizing Business Process Portal

You can localize your Business Process Portal interface to the language of a specific locale. Business Process Server provides out-of-box support for English. We also provide a basic level of translation (that is, many but not all parameters have been translated) to French, Japanese, Chinese and other languages. You can localize your interface for additional languages, provided the language is supported in Sun's J2SE Development Kit 5.0 (JDK) and J2SE Runtime Environment 5.0 (JRE). For a complete listing of locales (and languages) supported by Sun Java, refer to http://download.oracle.com/javase/1.5.0/docs/guide/intl/locale.doc.html.

**Note:** You cannot localize the **Login** and **Logout** pages. Business Process Portal displays **Login** and **Logout** pages using user's browser locale.

Decimal numbers are displayed in different formats in different languages. For example, in US English, period is used as the decimal separator, whereas comma is the decimal separator in French language. Business Process Portal validates and displays the decimal numbers based on the user's locale.

Business Process Portal does not support localization of decimal separator in the following pages. These pages accept and display decimal numbers in US English locale only.

- Form presentations with Business Objects
- Form presentations with Controls
- Reports
- All pages for filter management

**To localize Business Process Portal, complete the following steps:**

1. Specify appropriate date formats.
2. Translate the properties file.
3. Customize the default interface labels.

These steps are explained in the following sections.

Specifying the date formats

Currently Business Process Server supports 24 hour time format only for the locales other than English. The bpmportal.properties file contains the following keys for the date and time format.
You must update these values considering the following points.

- The value of the `dateFormat` key must be equal to the combination of values of the `Portal.DateOnlyFormat` and `Portal.TimeOnlyFormat` keys.
- The value of the `date_format` key must be equal to the value of the `Portal.DateOnlyFormat` key.

**Translating the Properties file**

The OEBPS_HOME\conf\properties folder contains the following Business Process Portal properties files, localized for their respective languages:

- `bpmportal_en.properties` (English)
- `bpmportal_es.properties` (Spanish)
- `bpmportal_fr.properties` (French)
- `bpmportal_ja.properties` (Japanese)
- `bpmportal_pt_BR.properties` (Portuguese)
- `bpmportal_zh.properties` (Chinese)

To create your localized properties file:

1. Create a copy of the `bpmportal.properties` file. Optionally, you can append the file name with the appropriate Locale ID (language or country code). For example, you can rename the properties file for the Korean language as `bpmportal_ko.properties`.

   **Note:** Java only supports Latin-1 or Unicode character sets. If the language you want to use is not compatible with Latin-1 or Unicode character sets, use the Java tool, native2ascii. For more information on using this tool, refer to [http://download.oracle.com/javase/1.5.0/docs/tooldocs/windows/native2ascii.html](http://download.oracle.com/javase/1.5.0/docs/tooldocs/windows/native2ascii.html).

2. Replace the English definitions in this file with the corresponding translations in your selected language.
3. Copy the translated properties file to `OEBPS_HOME\conf\properties` folder.

**Customizing the interface labels**

The common.jar file (located in OEBPS_HOME\lib folder) contains the `EBMSCommon_xxx` property that corresponds to the default interface labels.

To customize:
Adding support in Business Process Portal

Before localizing your Business Process Portal interface to your selected language, ensure that the option of selecting this language is made available to Business Process Portal users. You can specify this new language option in the supported languages parameter in the oebps.conf file (located in OEBPS_HOME\conf folder). To do so:

1. Open the oebps.conf file and navigate to the Business Process Server supported languages section (sample is shown below).

<pre>
&lt;param name="sbm.supportedlanguages">
  &lt;alias&gt;Supported Languages&lt;/alias&gt;
  &lt;visible&gt;false&lt;/visible&gt;
  &lt;description&gt;The languages currently supported&lt;/description&gt;
  &lt;legalvalues&gt;en_US|ja_JP|zh_CN|fr_FR&lt;/legalvalues&gt;
&lt;/param&gt;
</pre>

2. Append the Locale ID to the oebps.supportedlanguages parameter, separated by a comma. For example, to add support for the Korean language, modify the parameter as shown below,

\[ oebps.supportedlanguages=en_US,ko_KR \]

3. In case you have already started the Business Process Server, restart the server to reflect the changes.

Application users can now login to Business Process Portal and choose their language, using the Profile page. For information on changing user profile settings, refer to the "Managing User Profile" section of the Business Process Portal User's Guide.

---

Important: Business Process Portal displays the log messages using the language of the server's locale. When you view the log messages in Business Process Portal, it does not change the log messages' language to the language of your locale.

---

Localizing dashboard

You can localize the dashboard title and its description as well as widget title and widget description. Business Process Server reads the localized title and description for the dashboard and widgets from the OEBPS_HOME\conf\properties\bpmportaldashboard.properties file and displays them in the Business Process Portal. If Business Process Server does not find a parameter in this file, then it shows the user entered title and description.
Localizing dashboard widget title and description

To localize dashboard widget title and description:

1. Open the OEBPS_HOME\conf\properties\bpmportaldashboard_<locale>.properties file in a text editor, where <locale> is the target locale for localization.

2. Add the localized values for the title and description for the dashboard widget you want to localize in the properties file. The format for the key in the properties file is <widget title>.title for the widget title and <widget title>.desc for widget description, where <widget title> is the title you entered while adding the widget to the dashboard. While rendering the widget in the dashboard, Business Process Portal uses the localized value entered for title and description in the properties. If localized values are not present in the properties file, then Business Process Portal displays the user entered values. For example, if you enter "mywidget1" as title in the Add Component dialog box while adding the widget to the dashboard, then you need to add the following entries in the bpmportaldashboard_<locale>.properties file.

   mywidget1.title=My Sample Widget
   mywidget1.desc=Sample Widget

3. Save the file. Refresh Business Process Portal as explained in the Refreshing localized data on page 113 to reflect your updated data.

Localizing dashboard title and description

To localize dashboard title and description:

1. Open the OEBPS_HOME\conf\properties\bpmportaldashboard_<locale>.properties file in a text editor, where <locale> is the target locale for localization.

2. Add the localized values for the title and description for the dashboard you want to localize in the bpmportaldashboard_<locale>.properties file. The format for the key in the properties file is <dashboard name>.title for the dashboard title and <dashboard name>.desc for the dashboard description, where <dashboard name> is the name you entered while defining the dashboard. For example, if you specified "mydashboard" as the dashboard name in the Create Dashboard page while defining the dashboard, then you need to add the following entries in the bpmportaldashboard_<locale>.properties file.

   mydashboard.title=My Sample Dashboard
   mydashboard.desc=Sample Dashboard

3. Save the file. Refresh Business Process Portal as explained in the Refreshing localized data on page 113 to reflect your updated data.
Configuring dashboard widget height

You can specify the widget height while deploying it. You can change this widget height for a specific dashboard later after adding it to that dashboard. Business Process Server uses the text entered in the title field in the Add Component dialog box as a part of the key to look up the dashboard specific widget height in the bpmportaldashboard_<locale>.properties file.

**Note:** While rendering the widget in the dashboard, Business Process Server uses the title of the widget to lookup the height value specified for this dashboard widget. If it does not find height value, it uses the widget height specified while deploying the widget.

To change the dashboard widget height:

1. Open the OEBPS_HOME\conf\properties\bpmportaldashboard_<locale>.properties file in a text editor, where <locale> is the target locale for localization.
2. Add a parameter in the <title>.height format, where <title> the text entered in the title field at the time of adding the widget to the dashboard.
3. Set this parameter to a value specifying the widget height in pixels. For example, if you want to specify height as 400 pixels for a widget with title 'myTasks', then add the following entry in the bpmportaldashboard_<locale>.properties file.

   myTasks.height=400

**Note:** If you want to specify a different height for the widget in a different dashboard, you need to specify a different title for the widget for that dashboard to uniquely identify the widget height from the properties file.

4. Save the file. Refresh Business Process Portal as explained in the Refreshing localized data on page 113 to reflect your updated data.

Refreshing localized data

The bpmportaldashboard.properties file contains the title and description for the default dashboard and out-of-box widgets. However, you need to localize this file for other languages.

Ensure that the localized entries are copied into all locale specific dashboard properties files for all the locales in which you want to localize.

To refresh properties files without re-starting portal server:

1. From the Business Process Portal Administration tab, click System > Configuration > Portal. The Portal Configuration page is displayed.
2. Click Refresh.
Localizing Progress Developer Studio for OpenEdge

You can localize Progress Developer Studio for OpenEdge to a particular language of your choice. The following instructions demonstrate how to localize Progress Developer Studio for OpenEdge menus, titles of windows, labels of controls, messages and other displayed text in a specific language.

The `<Studio_Home\eclipse\plugins\com.savvion.studio.common <version><timestamp>\resources\i18n` folder (where `<version>` is the version of Progress Developer Studio for OpenEdge, and `<timestamp>` is the time when this version was generated) contains several localization property files, localized for their respective languages. For example:

- `designer_en.properties` (English)
- `designer_fr.properties` (French)
- `designer_ja.properties` (Japanese)
- `designer_zh.properties` (Chinese)

To localize Progress Developer Studio for OpenEdge:

1. Create a copy of any of the properties file and translate the listed definitions into the language of your choice.

   For example, to create a `designer.properties` file for the Korean language, create a copy of the `designer.properties` file and rename it as `designer_ko.properties`. Replace English definitions with definitions in the Korean language. For information on locales supported by Java, refer to http://java.sun.com/j2se/1.5.0/docs/guide/intl/locale.doc.html.

   **Note:** Java only supports Latin-1 or Unicode character sets. If the language you want to use is not compatible with Latin-1 or Unicode character sets, use the Java tool, native2ascii. For more information on using this tool, refer to native2ascii.

2. You do not need to localize the following `.properties` files: `bpserver.properties`, `bpmevents.properties`, `bpmwebflow.properties`, `bpmprocessstore.properties`, `bpmportal.properties`, `jasperreports.properties`, `oebps.properties`, `oepsemail.properties`, and `oebpslog.properties`.

3. After translating all the required `.properties` files, save the changes you made and restart Progress Developer Studio for OpenEdge to enable the new localizations.

Localizing applications

You need to perform the task of localizing Business Process Server applications separately. Progress Developer Studio for OpenEdge generates a properties file for every application containing Adapter worksteps or using the Form presentation format. This properties file namely, `<application_name>.properties`, is located in the properties folder in the `workspace\<application_name>` folder.
Note: To localize applications that do not include an Adapter workstep nor use the Form presentation format, you need to manually create localized values for UI elements including dataslot labels, menus, and any text seen by the user. Users can view these hard-coded values, localized to the selected language.

To localize the application’s properties file:

1. Create a copy of the `<application_name>.properties` file and translate the listed definitions into the language of your choice.
2. Rename the file with the appropriate Locale ID. For example, to localize `Approval.properties` file for the Korean language, create a copy of the file and rename it as `Approval_ko.properties`.

Localizing Business Process Modeler

Similar to how you localize Progress Developer Studio for OpenEdge, you can localize Business Process Modeler to a particular language of your choice. The following instructions demonstrate how to localize Business Process Modeler menus, titles of windows, labels of controls, messages and other displayed text into a specified language.

The `ProcessModeler\eclipse\plugins\com.savvion.processmodeler\conf\properties` folder contains the localization property files (namely, `designer_en.properties` (English), `designer_fr.properties` (French), `designer_ja.properties` (Japanese), and `designer_zh.properties` (Chinese)). If you are using a Business Process Modeler version prior to 7.0, the localization property files are located in the `ProcessModeler\conf\properties` folder.

To localize Business Process Modeler:

1. Follow the instructions provided in Steps 1–2, as described in Localizing Progress Developer Studio for OpenEdge on page 114.
2. After translating all the required `*.properties` files, save the changes you made and restart Business Process Modeler to enable the new localizations.

Localizing Jasper reports

Jasper reports do not support Japanese fonts by default. Therefore, Business Process Portal cannot display some layout elements like headers and footers. You must set the requisite font files in the classpath manually to resolve this issue.

To support Japanese fonts in Jasper reports:

1. Download the `itextasian-1.5.2.jar` or later from the URL [http://www.java2s.com/Code/Jar/i/Downloaditextasian152jar.htm](http://www.java2s.com/Code/Jar/i/Downloaditextasian152jar.htm) and copy it to the `OEBPS_HOME\lib` and `SBM_WebApp\WEB-INF\lib` folders.
2. In the `default.jasperreports.properties` file stored in the `OEBPS_HOME\conf\properties` and `SBM_WebApp\WEB-INF\classes\properties` folders, set the following two parameters to the values given below.
Customizing Jasper reports library

Dynamic theming for Jasper report library is supported in Progress OpenEdge 11.5 release. In Business Process portal, Jasper reports library is extensively used for the following out-of-the-box reports:

- Status Analysis report
- Time Analysis report
- Instance Analysis report
- Instance elapsed time report

**Note:** Migarte to Jasper reports library version 4.1.2.

For details, see the following topics:

- JRXML files
- Changing colors

**JRXML files**

In Business Process portal, the out-of-the-box reports are rendered using jrxm files, it is available under OEBPS_HOME\webapps\deploy\sbm.war\bpmportal\reports\jasper. It has the following tags:

- Page Header
OpenEdge has individual template for every theme. For the current release, it has five templates as follows:

- jr-style-theme01.jrtx
- jr-style-theme02.jrtx
- jr-style-theme03.jrtx
- jr-style-theme04.jrtx
- jr-style-theme05.jrtx

## Changing colors

You can change the front and back color of the tags listed:

To modify the color code:

1. Open the corresponding template file.
2. Modify the color code as required.
3. Refresh the page.

The modified styles are reflected.
This chapter describes how to customize OpenEdge for Ext JS 4.0. For details, see the following topics:

- Adding Ext JS 4.x files
- Customizing forms
Adding Ext JS 4.x files

If you are using any of the following JavaScript files from Ext JS 3.x in customized code, then replace these files with the corresponding new files from the same folder.

Table 18: Ext JS files to be replaced for backward compatibility

<table>
<thead>
<tr>
<th>Ext JS 3.x files</th>
<th>Ext JS 4.x files</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Folder:</strong> &lt;SBM_WebApp&gt;/deploy/sbm.war/bpmportal/javascript</td>
<td><strong>Folder:</strong> &lt;SBM_WebApp&gt;/deploy/sbm.war/bpmportal/javascript</td>
</tr>
<tr>
<td>• bmutil.js</td>
<td>• bmutil_4x.js</td>
</tr>
<tr>
<td>• BmViewport.js</td>
<td>• BmViewport_4x.js</td>
</tr>
<tr>
<td>• top.js</td>
<td>• top_4x.js</td>
</tr>
<tr>
<td><strong>Folder:</strong> &lt;SBM_WebApp&gt;/deploy/sbm.war/bpmportal/javascript/bm/common</td>
<td><strong>Folder:</strong> &lt;SBM_WebApp&gt;/deploy/sbm.war/bpmportal/javascript/bm/common</td>
</tr>
<tr>
<td>• bmdsfilter.js</td>
<td>• bmdsfilter_4x.js</td>
</tr>
<tr>
<td>• bmfield.js</td>
<td>• bmfield_4x.js</td>
</tr>
<tr>
<td>• bmfilter.js</td>
<td>• bmfilter_4x.js</td>
</tr>
<tr>
<td>• datetime.js</td>
<td>• datetime_4x.js</td>
</tr>
<tr>
<td><strong>Folder:</strong> &lt;SBM_WebApp&gt;/deploy/sbm.war/bpmportal/javascript/ux/superboxselect</td>
<td><strong>Folder:</strong> &lt;SBM_WebApp&gt;/deploy/sbm.war/bpmportal/javascript/ux/superboxselect</td>
</tr>
<tr>
<td>• SuperBoxSelect.js</td>
<td>• SuperBoxSelect_4x.js</td>
</tr>
</tbody>
</table>

Customizing forms

The current version of Progress Developer Studio for OpenEdge uses Ext JS version 4.x. However, forms designed using the previous version of Progress Developer Studio for OpenEdge work seamlessly in the current version of Progress Developer Studio for OpenEdge.

Ext JS 4.x files are stored in the <SBM_WebApp>/deploy/sbm.war/bpmportal/javascript/ext4x folder. You must update all your customized JSPs to point to this folder to take advantage of Ext JS 4.x.

Note that Business Process Portal uses the Cascading Style Sheets (CSS) stored in the <SBM_WebApp>/deploy/sbm.war/bpmportal/javascript/ext4x/resources/css folder.

For more information about Ext JS, you can refer to Sencha Inc.’s site http://www.sencha.com.
Single Sign-on for Business Process Server

This chapter explains Single Sign-on (SSO) for Progress OpenEdge Business Process Server and it shows you how to use the Connect() and GetClientPrincipal() methods of the Progress.BPM.UserSession class for an OpenEdge client to log into the Business Process Server using the SSO mechanism.

Before connecting to Business Process Server using the SSO mechanism, you must configure single point of authentication in Business Process Server. Also, you must configure both the single point of authentication and SSO with the same domain for user authentication. For more information on configuring single point of authentication, see the OpenEdge Business Process Server: Business Process Portal Administrator's guide.

For details, see the following topics:

- SSO support for Business Process Server
- Domain Management
- Example of OpenEdge client-side code

SSO support for Business Process Server

Progress OpenEdge Business Process Server supports single sign-on (SSO). Using the SSO mechanism a user who is authenticated in an OpenEdge application can also sign into Business Process Server without the need for re-authentication. SSO is an access control feature for multiple related systems. The SSO mechanism validates a user's identity to sign into one system if the user has been authenticated by another system within the same security domain.
For more information about user authentication using SSO, see the OpenEdge Getting Started: Identity Management guide.

The ABL methods, `Connect()` and `GetClientPrincipal()`, enable a user of an OpenEdge client to perform a SSO into the Business Process Server using a sealed `Client-Principal` object as an identity token. These methods are public methods of the `Progress.BPM.UserSession` class. Each `Client-Principal` object is an ABL security token. A security token contains credentials that is used to establish a user’s identity for an ABL session. For more information on using the ABL methods of the `Progress.BPM.UserSession` class, see the OpenEdge Development: ABL Reference guide.

For SSO into the Business Process Server, an OpenEdge client must do the following:

- Use the `Register-Domain()` method on the `SECURITY-POLICY` system handle to create and load a domain into the session registry. The default registry supports only the "bpm" domain and the access code "oebpm".


- Connect to the Business Process Server using the `UserSession:Connect(hCP)` method, where `hCP` is a handle to the `Client-Principal` object.

- The `Client-Principal` object specified in the `Connect(handle)` method must contain the credentials of the user who has been authenticated by an OpenEdge application.

- The `Client-Principal` object must specify a domain ID of the current domain registry. The default registry supports only the "bpm" domain.

- The `Client-Principal` object must be sealed with a domain access code. The "bpm" domain specified in the default domain registry requires the access code "oebpm".

**Note:** The authenticated user identified by the sealed client-principal need not be defined as a valid user in the Business Process Server authentication system such as in the JDBC or OEHybrid realms.

### Domain Management

An `OEDomainRegistry` object is used to define a list of domains whose members may access the Business Process Server. The `OEDomainRegistry` consists of a set of domain names along with their corresponding domain access codes. For an ABL application to connect a `UserSession` object using a client-principal handle, the domain specified in the `Client-Principal` object method must be defined in the active BPM domain registry. The domain ID is a required parameter that is specified in the `Client-Principal` object.

The domain to which SSO access is allowed by Business Process Server is specified in the default BPM domain registry, as mentioned in the following table:

<table>
<thead>
<tr>
<th>Domain name</th>
<th>Domain access code</th>
</tr>
</thead>
<tbody>
<tr>
<td>bpm</td>
<td>oebpm</td>
</tr>
</tbody>
</table>
Using the Domain Registry utility

Other than the default domain registry, the domains supported by Business Process Server can be specified in an encrypted domain registry file. At run time, the Business Process Server loads the domain registry file from the OpenEdge working directory $WRKDIR and allow connections only from users in the specified domains. You can create the domain registry file using the Generate Domain Registry gendomreg.bat utility, which is located in the OpenEdge installation directory $DLC\bin.

**Note:** At run time, if the domain registry file is not found, then the default BPM domain registry is used to ensure that the specified domain is defined in the domain registry.

To use the Generate Domain Registry tool:

1. Create a list of allowed domain names and their corresponding access codes in a Comma Separated Value (CSV) text file. Each domain and its access code must be specified on a separate line, with the domain name and access code separated by a comma.

   The following is an example of a CSV file:
   
   ; This is a BPM domain registry CSV file
   ; lines that start with a ; are comments
   ; since this registry supersedes the default registry
   bpm , oebpm
   oedom1 , dac1
   oedom2 , dac2

   **Note:** The above example illustrates that there are two domains added to domain registry in addition to the default bpm domain.

2. Save the Comma Separate Value file. For example, save the CSV file with the name as bpm_domain_registry.csv.

3. From proenv prompt, run the Generate Domain Registry tool to generate an encrypted domain registry file.

   For example, generate the encrypted domain registry file as bpm_domain_registry.bin.

   a) Click Start > Programs > Progress > Proenv. The proenv command prompt opens.
   b) Type the following and then press ENTER:

   ```
   proenv>gendomreg bpm_domain_registry.csv bpm_domain_registry.bin
   ```

4. Copy the generated domain registry file into the OpenEdge working directory ($WRKDIR):

   ```
   proenv>copy bpm_domain_registry.bin <working_directory_path>\BPMDomainRegistry.bin
   ```

   **Note:** As per step 4, you must copy the domain registry file with the name BPMDomainRegistry.bin. At run time, the Business Process Server loads the BPMDomainRegistry.bin file from your working directory and allows connections only from users in the specified domains.
Example of OpenEdge client-side code

The following example illustrates an OpenEdge client-side code implementation to support SSO for user authentication into the Business Process Server. The code block uses the Connect( ) and GetClientPrincipal( ) methods added to the Progress.BPM.UserSession class for user authentication using a Client-Principal object.

For more information on these methods of the Progress.BPM.UserSession class, see the OpenEdge Development: ABL Reference guide.

```abl
DEF VAR plOK AS LOG NO-UNDO.
DEFINE VARIABLE pURL AS CHAR INITIAL "-URL SBMServerDC://<localhost>:18793/".
DEFINE VARIABLE pUserSession AS Progress.BPM.UserSession.
DEFINE VARIABLE retStr AS CHARACTER.
DEFINE VARIABLE tmpStr AS CHARACTER.
DEFINE VARIABLE hCP AS HANDLE NO-UNDO.
DEFINE VARIABLE dac AS CHARACTER INITIAL "oebpom".

OUTPUT TO bpm_client_sealed.out APPEND.

PUT UNFORMATTED "-- bpm_client_sealed.p start -------------------------" SKIP.

/* create a domain registry in code */
IF plOK
  THEN DO:
    plOK = SECURITY-POLICY:LOCK-REGISTRATION().
    IF plOK
      THEN PUT UNFORMATTED "domain registry OK." SKIP.
      ELSE DO:
        PUT UNFORMATTED "lock-registration() failed." SKIP.
        QUIT.
    END.
  ELSE DO:
    PUT UNFORMATTED "register-domain() failed." SKIP.
    QUIT.
  END.

/* create client principal */
CREATE CLIENT-PRINCIPAL hCP.
  hCP:INITIALIZE("oebpomuser@bpm", "12345", ?, "oebpomuser").
plOK = hCP:SEAL(dac).

/* create BPM UserSession */
PUT UNFORMATTED "before CONNECT() : hCP= " hCP
  " session-id= " hCP:SESSION-ID
  " login-state= " hCP:LOGIN-STATE
  SKIP.

pUserSession = NEW Progress.BPM.UserSession(pURL).
PAUSE.

/* connect to BPM server using CP as credentials */
plOK = pUserSession:Connect(hCP).
IF plOK THEN
  PUT UNFORMATTED "Connected to BPM Server OK." SKIP.
ELSE DO:
```

Chapter 14: Single Sign-on for Business Process Server

OpenEdge Business Process Server: Customization Guide
PUT UNFORMATTED "Connect to BPM Server FAILED." SKIP.
QUIT.
END.

PUT UNFORMATTED "after CONNECT() : hCP= " hCP
   " session-id= " hCP:SESSION-ID
   " login-state= " hCP:LOGIN-STATE
SKIP.

plOK = hCP:VALIDATE-SEAL(dac).

IF plOK THEN
   PUT UNFORMATTED "sealed CP received from BPM Server is VALID." SKIP.
ELSE DO:
   PUT UNFORMATTED "sealed CP received from BPM Server is INVALID." SKIP.
      pUserSession:Disconnect(TRUE).
      QUIT.
END.

IF (plOK)
   THEN DO:
      PAUSE.
      hCP = pUserSession:GetClientPrincipal().
      PUT UNFORMATTED
         "after GetClientPrincipal() : hCP= " hCP
         " session-id= " hCP:SESSION-ID
         " login-state= " hCP:LOGIN-STATE
      SKIP.
   END.

IF (plOK)
   THEN DO:
      pUserSession:Disconnect(TRUE).
      PUT UNFORMATTED "Disconnected from BPM Server OK." SKIP.
   END.

DELETE OBJECT hCP.

PUT UNFORMATTED "-- bpm_client_sealed.p end -------------------------" SKIP.
OUTPUT CLOSE.
QUIT.
Glossary

ACL manager
In Business Process Server, Access Control List Manager provides a finer, more precise control over user access rights for resources and actions.

Activity workstep
In Business Process, the basic unit of work; must be performed by one or more human performers (valid individual user, multiple users or user group).

Adapter
A Java class that integrates remote, third party classes and actions with Business Process. An adapter can automate certain functions and tasks performed by a remote server or other external systems.

Administration
A module in Business Process Portal enabling the administrator to perform tasks such as installing/uninstalling applications, modifying configuration parameters controlling Business Process operations, and manage users, groups and access control. The Administration module is visible only to application users who have permissions to access it.

Application
In Business Process, an application is an installed, executable business process that automates a business flow.

Balanced scorecard
A management application in the Management module that measures performance by analyzing how an organization’s business activities help it achieve its strategic goals. The Balanced Scorecard provides an analysis from a range of perspectives.
BAM

Business Activity Management combines Business process management with strategic and analytical information on specific business performance indicators, providing real-time status information and identifying critical events to assist senior management in making informed business decisions.

BPM Events

A Business Process Server component that provides an open event-driven rule engine to formulate and enforce policies in business applications.

BPM Webflow

A Business Process Server component that enables users to develop customizable, sophisticated presentation flows for business processes, install them as Web applications, and execute them on their Web browsers.

BPEL

BPEL (Business Process Execution Language) for Web services is an XML-based language designed to enable task-sharing for a distributed computing or grid computing environment - including across multiple organizations - using a combination of Web services.

BPMN

BPMN (Business Process Modelling Notation) provides businesses with the capability of defining and understanding their internal and external business procedures through a Business Process Diagram giving organizations the ability to communicate these procedures in a standard manner.

BP Server

A Business Process Server component that provides a flexible, lightweight, scalable workflow process engine for intranets, extranets, and the Internet.

Business calendar

A Business Process Server feature that accurately calculates the Due Date of tasks, and provides support for multiple business calendars across different time zones.

Business flow

The logical sequence of process activities, related to one another by a triggering activity, to achieve an outcome. It represents a business process that begins with a commitment and ends with the termination of that commitment. In Business Process Server, business flow includes Workflow (the flow of all human-performed activities), integration flow (the flow of activities performed by systems) and presentation flow (from a user’s viewpoint, the flow of data from one Web page to the next).

Business logic

The control flow and information flow among worksteps that define a business process.

Business object

A representation of an activity in the business domain, including its name, definition, attributes, behavior, relationships and constraints.

Business process

A process involving multiple worksteps in the form of operations, interactions and notifications performed by a user, group of users, an external adapter, or a script.
Business Process Server application

An application is an implementation of a business process. It can contain one or more process templates, performers, adapters, customized forms or rules. An application can be published, installed and run on BP Servers. In Business Process Server, an application is an installed, executable business process that automates a Workflow.

Business Process Server Web services

A Business Process Server component that allows application developers to; a) publish their applications as Web services, and b) find and convert other available Web services on the Internet into Business Process Server applications.

Business Process Portal

A Business Process Server component that offers users, managers, administrators and developers a unified, customizable portal for single sign-on access to all Business Process Server functionalities to which they are granted permission.

Business process management

The concept of guiding work activities through a multi-step business process in order to improve performance and reduce costs within and across functional business units.

Business Process Modeler

A stand-alone component that enables users to design templates for basic business processes.

Business rule

A combination of elements, including validation edits, logon verifications, database lookups, policies and transformations, that represent an enterprise’s way of doing business.

Control flow

The sequences of worksteps and workstep conditions, as defined in a process template in Progress Developer Studio for OpenEdge or Business Process Modeler.

Dashboard

A Business Process Server feature that provides a graphic overview of the status of several business processes on a single Web page, enabling users to monitor the progress of each process. Users can view business processes across all applications or for a selected application.

Dataslot

A data placeholder that persists through the entire process and defines the information flow of the business process. Dataslots are associated with processes, where they can add information into (Input type) or out of (Output type) worksteps, and appear as editable or read-only fields on a user’s interface.

Expression editor

A Business Process Server tool that enables users to define complex conditional expressions within a Decision gateway to support their business requirements.

Group

In Business Process Server, an entity that has as members valid users or other groups who perform related work and have authorized access to specific components.
Heatmap
A Business Process Server feature that provides a convenient, graphical tool for managers to visually locate the bottlenecks in the process execution. It helps managers to get an overview of the status of the currently active instances, identify suspended instances, and analyze the history of the completed instances.

Home
A module in Business Process Portal through which users interact with Business Process Server. Using the Home module, users complete entries to various tasks and applications, update profile, set preferences, and link to the support infrastructure required to achieve these tasks. The Home module is the primary interface for application users.

Infopad
In Business Process Server, a data structure used to capture business metrics, typically displayed as a table with one or two dimensions.

Instance
An individual object within a specific class. In Business Process Server, a self-contained unit that is created each time you use a process template to run a Business Process Server application.

KPI
Key Performance Indicator, used in the Balanced Scorecard system, that provides the data translating enterprise goals into a set of measurable objectives.

Managed Adapter
In Business Process Server, a Managed Adapter is an implementation of an adapter interface that facilitates data exchange between Business Process Server processes and external applications.

Management
A module in Business Process Portal enabling the managers to query, report, and control processes and resources for application users. The Management module is visible only to application users who have permissions to access it.

Migration
The process of moving from the use of one operating environment to another operating environment that is typically seen as improvement. Migration can involve moving to new hardware, new software, or both. It may involve a new application, another type of database, or a redesigned network. Migration is also used to refer simply to the process of moving data from one storage device to another. Business Process Server supports data migration as well as application migration.

Performer
An entity that executes a workstep. Depending on the workstep type, the performer can be a human user, a group of users, an adapter or other external performer, or a script.

Presentation flow
The flow of information and user input from one interface to the next. Typically related to a single Activity workstep in the process and generated in a BPM Webflow environment.

Process engine
Orchestrates the execution of business processes and also coordinates conversations among process engines based on public processes, which forms the backbone of global business collaboration.
Glossary

Process refresh
A Business Process Server feature for replacing the installed process without versioning, facilitating the running process instances to refresh and seamlessly adapt to the new Workflow.

Process template
In Business Process Server, a model of business flow that includes worksteps, connectors and dataslots. After users publish and install it as an application in Business Process Server folder structure, they can use the application to create process instances.

Progress Developer Studio for OpenEdge
An Integrated Development Environment for Business Process Server that enables application users to develop and publish a Business Process Server application without leaving the development environment.

Role
The actions and activities assigned to a valid application user who is a member of a group. In Business Process Server, only members of a group can be assigned a role. A role indicates the relationships of the user in a group context.

Rollback
In Business Process Server, a feature that restarts the Workflow from a workstep previously selected as the rollback point in the process, performed automatically in the event of a failure.

Rule wizard
An interactive utility that enables application users to quickly develop rules that can be applied to a business process.

Swim lanes
Used in Workflow diagrams to organize complex processes across functional boundaries. For example, seen as horizontal lines on a process map, swim lanes can be used to place individual task steps into different categories that depend on task ownership.

Task
In Business Process Server, a performer is assigned one or more work items that the performer sees as tasks. There are two types of tasks: Assigned, which are assigned specifically to you; and Available, which are available to be performed by you or other members of your user group.

User
In Business Process Server, a valid human performer with authorized access to specific modules.

Workflow
The logical sequence of activities performed by human performers. Workflow includes the tasks, procedural steps, organizations or people involved, required input and output information, and tools needed for each activity in a business process.