OpenEdge Business Process Server: Managed Adapters 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: Introducing Managed Adapters**

- About Managed Adapters .......................................................... 15
- Basic concepts of Managed Adapters .......................................... 16
- Managed adapter features ......................................................... 17

**Chapter 2: Understanding the Managed Adapter framework**

- Framework components .......................................................... 20
  - Interaction of the framework components ................................ 20
  - File formats used in the Managed Adapter framework ............... 22
  - Supported data types ............................................................ 27
  - Supported APIs ..................................................................... 32
  - Adapter logging .................................................................... 34
- Using the Adapter Configurator and Map Configurator .................. 35
- Deploying maps using MapDeployer ........................................... 37

**Chapter 3: Database Managed Adapter**

- Database Adapter features ......................................................... 40
- Database Adapter components .................................................... 40
- Working with the Database Adapter ............................................ 41
  - Configuring the Database Adapter .......................................... 41
  - Using stored procedures ....................................................... 47
  - Using the Expert mode .......................................................... 48
- Working with DataDirect Cloud drivers in the Database Adapter .... 52
- Using the Database Adapter ....................................................... 52
- Updating Database properties and server files .............................. 53
- Using the DBAdapterDemo sample application ............................ 55
- Tutorial 1: Using the Database Adapter ...................................... 55
  - Step 1: Creating the customer database table .......................... 55
  - Step 2: Creating the process flow .......................................... 56
  - Step 3: Defining the dataslots ............................................... 56
Chapter 7: JMS Managed Adapter

JMS Adapter features .................................................................99
How the JMS Adapter works.......................................................101
Working with the JMS Adapter....................................................102
  Assigning JMS Adapter as workstep performer.......................102
  Configuring the JMS Adapter..................................................102
  Defining mapping with the Mapping Configurator....................104
  Receiving messages: Passing the CorrelationID .....................104
Tutorial 1: Using the JMS Adapter in Send-Only mode.................105
  Defining the process workflow for the Send-Only mode............105
  Configuring the JMS Adapter for Send-Only mode....................106
  Defining mapping for the Send-Only mode..............................107
  Troubleshooting for the Send-Only mode...............................107
Tutorial 2: Using the JMS Adapter in Send/Receive mode.............108
  Defining the process workflow for the Send/Receive mode.........108
  Configuring the JMS Adapter for the Send/Receive mode..........109
  Defining mapping for the Send/Receive mode.........................109
  Defining the SendReceive workstep as asynchronous for the Send/Receive mode....110
  Deploying and testing the JMS Adapter for the Send/Receive mode...110
  Troubleshooting for the Send/Receive mode..........................110
  Advanced exercises for the Send/Receive mode.......................111
Tutorial 3: Using the JMS Adapter in Send-Only and Receive-Only mode 111
  Defining the process workflow for the Send-Only and Receive-Only mode.................111
  Configuring the JMS Adapter for the Send workstep.................112
  Mapping the send workstep dataslots...................................113
  Configuring the JMS Adapter for the Receive workstep...............113
  Mapping the Receive workstep dataslots................................113
  Defining the Receive workstep as asynchronous for a Receive-Only mode.................114
  Deploying and Testing the JMS Adapter for Send-Only and Receive-Only modes........114
  Troubleshooting for Send-Only and Receive-Only modes...........114
Tutorial 4: JMS Adapter - Receiving messages from external applications 114
  The Order Processing application........................................115
  Defining the process workflow for receiving messages from external applications........115
  Configuring the JMS Adapter to receive messages from external applications.............116
  Defining mapping to receive messages from external applications.........................116
  Troubleshooting for receiving messages................................116

Chapter 8: OpenEdge Managed Adapter

OpenEdge Adapter features.....................................................119
Working with the OpenEdge Adapter........................................120
  Configuring the OpenEdge Adapter.......................................120
  Defining the dataslot mapping..............................................127
Chapter 9: Sonic ESB Managed Adapter.........................................................135
  Sonic ESB Adapter features.........................................................................................135
  Prerequisites..................................................................................................................136
  Overview.......................................................................................................................136
  Mapping OpenEdge ABL dataslots to Sonic ESB message.............................................136
  Custom properties.........................................................................................................137
  Working with Sonic ESB Adapter...................................................................................137
  Configuring the Sonic ESB Adapter................................................................................138
  Defining the dataslot mapping.......................................................................................141
  Defining the Advanced properties................................................................................141

Chapter 10: Corticon Managed Adapter.............................................................143
  Working with Corticon Adapter.....................................................................................143
  Configuring the Corticon Adapter................................................................................144
  Defining the Dataslot Mapping......................................................................................145

Chapter 11: SharePoint Managed Adapter.......................................................147
  Working with SharePoint Adapter................................................................................148
  Configuring the SharePoint Adapter............................................................................148
  Defining the Dataslot Mapping......................................................................................154
  Using SharePoint Adapter as an Adaplet.....................................................................154

Chapter 12: Web Service Managed Adapter....................................................157
  Overview.......................................................................................................................157
  Introducing Web Service Adapter..............................................................................158
  Using the Web Service Adapter..................................................................................159
  Configuring the Web Service Adapter.......................................................................159
  Fault handling.............................................................................................................171
  Describing a fault........................................................................................................171
  Fault handling in Adapter Configurator.......................................................................171
  Mapping an XPath expression to a user-defined parameter.......................................172
  Dataslot mapping for fault handling...........................................................................172

Glossary.......................................................................................................................175
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>
<tbody>
<tr>
<td>Application User</td>
<td><em>Business Process Portal User’s Guide</em></td>
</tr>
<tr>
<td></td>
<td><em>First Steps Guide</em></td>
</tr>
<tr>
<td></td>
<td><em>Terminology Guide</em></td>
</tr>
<tr>
<td>Manager</td>
<td><em>Business Process Portal Manager’s Guide</em></td>
</tr>
<tr>
<td></td>
<td><em>Business Process Portal User’s Guide</em></td>
</tr>
<tr>
<td></td>
<td><em>Terminology Guide</em></td>
</tr>
<tr>
<td>If you are the …</td>
<td>Read the …</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Application Developer    | Application Developer’s Guide  
BP Server Developer’s Guide  
BPM Events User’s Guide  
Business Process Portal Manager’s Guide  
Business Process Portal User’s Guide  
OpenEdge Getting Started: Developing BPM Applications with Developer Studio  
Customization Guide  
Managed Adapters Guide  
First Steps Guide  
Terminology Guide  
Server Administrator’s Guide  
Web services Developer’s Guide |
| Business Process Server Administrator | BPM Events User’s Guide  
Business Process Portal Administrator’s Guide  
Business Process Portal Manager’s Guide  
Business Process Portal User’s Guide  
OpenEdge Getting Started: Installation and Configuration Guide  
Managed Adapters Guide  
Terminology Guide  
Server Administrator’s Guide  
Troubleshooting Guide for Administrators |

For the latest Business Process Server documentation updates, see OpenEdge Product Documentation on PSDN (http://communities.progress.com/pcom/docs/DOC-16074).

## 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><em>file path</em></td>
<td>Indicates folder paths and file names.</td>
</tr>
</tbody>
</table>
### Purpose

<table>
<thead>
<tr>
<th>Convention (styles and terms)</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>italic</em></td>
<td>Indicates book titles.</td>
</tr>
<tr>
<td><em>monospace</em></td>
<td>Represents code segments or examples.</td>
</tr>
<tr>
<td>Backward slash &quot;&quot;</td>
<td>Indicates the path in Windows environment. For UNIX environment, replace with forward slash &quot;/&quot;</td>
</tr>
<tr>
<td>OEBPS_HOME or %OEBPS_HOME%</td>
<td>Represents the installation folder of Business Process Server, C:\Progress\OpenEdge\oebpm\server.</td>
</tr>
<tr>
<td>STUDIO_HOME or %STUDIO_HOME%</td>
<td>Represents the installation folder of OpenEdge BPM components, C:\Progress\OpenEdge\oebpm\studio.</td>
</tr>
<tr>
<td>JBOSS_HOME or %JBOSS_HOME%</td>
<td>Represents the installation folder of JBOSS server, C:\Progress\OpenEdge\oebpm\jboss.</td>
</tr>
</tbody>
</table>

### 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 the following table.

**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></td>
<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>
</tr>
<tr>
<td>Telephone¹</td>
<td>1-781-280-4999 for US, Latin America and Canada</td>
</tr>
<tr>
<td></td>
<td>1-781-280-4543 for the Product Support Fax Line</td>
</tr>
<tr>
<td>Postal Address¹</td>
<td>Progress Software Corporation</td>
</tr>
<tr>
<td></td>
<td>14 Oak Park Drive</td>
</tr>
<tr>
<td></td>
<td>Bedford, MA 01730, USA.</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.
Introducing Managed Adapters

In a typical business scenario, business processes may span across multiple applications having diverse data structures, formats, and protocols. Adapters help applications exchange information and participate in distributed business processes by translating the data from one application to a format that the other application understands.

Adapters offer high flexibility to exploit fast changing technology while continuing to exchange information with legacy applications. Adapters also eliminate custom coding and provide quick, efficient access to business information across an organization.

For details, see the following topics:

• About Managed Adapters
• Basic concepts of Managed Adapters
• Managed adapter features

About Managed Adapters

Business Process Server provide the following predefined Managed Adapters.

Table 2: Predefined Managed Adapters

<table>
<thead>
<tr>
<th>Adapter category</th>
<th>Adapter name</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBAdapters</td>
<td>GenericDBAdapter</td>
</tr>
</tbody>
</table>
### Basic concepts of Managed Adapters

**Figure 1: Managed Adapter basic operations** on page 16 shows the basic operations of a Managed Adapter.

When a business process reaches a Managed Adapter workstep, it invokes its adapter, which in turn exchanges data with the enterprise application. For each individual enterprise application, a specific adapter is required.
The Managed Adapters we provide act as a translating module that converts the business process-specific protocol to another application-specific protocol. In this usage, a Managed Adapter is a pluggable component that connects your business process to an external system such as a database or ERP system, or performs its own function of data transformation. Part of the functions related to the operation of the Managed Adapters (such as adapter configuration, input/output mapping) are handled by standardized in-house components, simplifying the development process of such adapters.

Managed adapter features

Managed Adapters support the following features:

- **Adapter Framework**
  - Configuration and mapping framework.
  - Better and extendable mapping.
  - Customizable GUI.
  - Ability to define complex mapping.

For more information, refer to Understanding the Managed Adapter framework on page 19

- **Database Adapter**
  - Seamless integration with Progress Developer Studio for OpenEdge.
  - Sequential execution of multiple SQL statements.
  - Support for Business Process Portal customization.

For more information, refer to Database Managed Adapter on page 39

- **E-mail Adapter**
  - Sending multipart MIME e-mail messages.
  - Generating messages that contain parts in all formats supported by the standard format plug-ins.
  - Sending messages to multiple recipients in a single workstep, while customizing the message content for each recipient.
  - Using the plug-in development kit to customize formats.

For more information, refer to Email Managed Adapter on page 63

- **File Adapter**
  - Facilitates reading from and writing to a file.
  - Supported file formats - text, delimited text, and XML.

For more information refer to File Managed Adapter on page 77

- **FTP Adapter**
  - Transfers files between an FTP server, a local file system.

For more information, see FTP Managed Adapter on page 95
Chapter 1: Introducing Managed Adapters

- **JMS Adapter**
  - Publishes and subscribes to JMS messages from messaging systems including MQ Series.
  - Supported modes - Send-only, Receive-only, Send / Receive.
  
  For more information refer to JMS Managed Adapter on page 99.

- **OpenEdge Adapter**
  - Integrates Business Process Server applications with Progress OpenEdge® solutions.
  - Interacts with OpenEdge procedures using BIZOE (Progress Interface Definition language) files.
  
  For more information refer to OpenEdge Managed Adapter on page 119.

- **Sonic ESB Adapter**
  - Interacts with Sonic® ESB.
  - Invokes a Sonic ESB process or service, or sends a message to Sonic ESB endpoint.
  
  For more information refer to the Sonic ESB Managed Adapter.

- **Web Service Adapter**
  - Enables you to find (or subscribe to) the services of other businesses on the Web.

**Note:** You can also access iWay adapters through the Web Service Adapter.

For more information, see Web Service Managed Adapter on page 157.

- **Sharepoint Adapter**
  - Invokes a Sharepoint Process or service, or sends a message to SharePoint.
  
  For more information, see SharePoint Managed Adapter on page 147.

For more information on common business process- and adapter-related terms, see the Glossary section. For the complete list of Business Process Manager terminology, see the OpenEdge Business Process Server: Terminology Guide.
Understanding the Managed Adapter framework

Progress Developer Studio for OpenEdge and Business Process Modeler provides a group of predefined Managed Adapters that you can use to exchange information between your business processes and other popular applications. A Managed Adapter is a translating module that converts the process-specific protocol to another application-specific protocol.

When a Managed Adapter is used in a workstep, the Application Developer can define complex mapping between your process dataslots and adapter inputs/outputs in an external application. At run time, when the given workstep is executed, the Managed Adapter uses a mapper to interpret the mapping and configuration information that was defined at design time. This run-time mapper sets the adapter inputs and configuration, and maps the outputs to the appropriate output dataslots after the adapter execution.

The Configuration and Mapping Framework enables you to carry out the configuration and dataslot mapping for each Managed Adapter. The framework offers:

- Improved, extendable mapping.
- Customizable GUI for adapter configuration and dataslot mapping.
- Ability to define complex mapping during process design.

The next section describes this framework. The subsequent section, Using the Adapter Configurator and Map Configurator on page 35, provides an example in using the framework to configure and map a managed adapter.

For details, see the following topics:

- Framework components
- Using the Adapter Configurator and Map Configurator
Framework components

The Adapter Configuration and Mapping Framework is now automated, and you do not need to use command-line options. However, if you want to use the command-line tools, these options, which are located in the OEBPS_HOME\bin folder, are briefly described below:

- **Adapter Installer**: This command-line tool creates the initial file describing a Managed Adapter.

- **Adapter Configurator**: This GUI component generates a configuration object providing workstep-specific adapter configuration parameters, inputs, and outputs.

- **Adapter Deployer**: This tool imports (or deploys) a predefined and preconfigured adapter in a BP Server or Web application.

- **Map Configurator**: This GUI component enables you to specify mapping between adapter inputs/outputs and the dataslots in your business process.

- **Map Deployer**: This tool deploys the maps generated by the MapConfigurator, and is automatically invoked if the process is published through Progress Developer Studio for OpenEdge or Business Process Portal.

- **Run-time Mapper**: This map tool interprets maps generated by the MapConfigurator and maps dataslots to adapter inputs, and adapter outputs back to dataslots.

The GUI components have default implementations allowing simple adapters to be developed without requiring an additional effort to build a custom GUI. A complex adapter may provide custom implementation for one or more of these components. The custom implementation facilitates seamless integration for complex GUIs, as well as complex mapping, if any.

Interaction of the framework components

The interaction of the framework components is illustrated in Figure 2: Interaction between components of the Managed Adapter framework on page 21. The sequence of this interaction is described below, with each step in the sequence referring to a number in Figure 2: Interaction between components of the Managed Adapter framework on page 21.

1. **Adapter Configurator** is an adapter-specific GUI component that is used to build the adapter configuration. The Adapter Configurator 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 config parameters such as temporary directories, or user names and passwords.

   - The Adapter Configurator GUI is automatically started from Progress Developer Studio for OpenEdge (or Business Process Modeler) by default, but it can also be started through a standard command-line tool (AdapterConfigurator.cmd/AdapterConfigurator.sh) by providing the category and name of the adapter. The command-line tool (see 1a in Figure 2: Interaction between components of the Managed Adapter framework on page 21) reads the Adapter Definition file, finds the configurator GUI class name, and calls the Adapter Configurator. The Adapter Configurator may also be a standalone program, not executed through the command-line tool, but called directly.

2. The Adapter Configurator stores the configuration in the config.xml file. Refer to config.xml on page 22 for a description of the file. The Adapter Definition file contains mapping data in the mapping.prop file. Refer to mapping.prop on page 24 for a description of this file. Both the files, mapping.prop and config.xml, are stored in the Adapter Instance Directory and are not yet associated to a specific BP Server Application or a Web application.

3. **Adapter Deployer**: This tool deploys the preconfigured adapter in a BP Server or Web application.
4. **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. By default, the Map Configurator GUI is automatically started by Progress Developer Studio for OpenEdge (or Business Process Modeler).

![Figure 2: Interaction between components of the Managed Adapter framework](image)

5. **Map Deployer.** When a process is ready to be published, the maps and adapter definitions need to be deployed. The deployment of maps and adapter definitions indicates that the data stored up to that point in various files are 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 automatically by Progress Developer Studio for OpenEdge or Business Process Portal. Alternatively, you can use the MapDeployer command-line tool.

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

6. **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, the Run-time Mapper interprets the input and output maps stored in the database repository, converts the Progress Developer Studio for OpenEdge 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. 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.
File formats used in the Managed Adapter framework

The following sections describe the format of the files used by the Adapter Mapping and Configuration Framework and provide examples of file formats. For information on the `config.xml` file, see the next section. For information on the `mapping.prop` file, see `mapping.prop` on page 24. For information on the `input.map` and `output.map` files, see `input.map` and `output.map` on page 25. For information on supported Data Types, see Supported Data Types section in the Progress OpenEdge Business Process Server: Managed Adapters Guide. For information on supported APIs, see the Supported APIs section in Progress OpenEdge Business Process Server: Managed Adapters Guide.

**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 any other custom format.

**Location**

This file is stored in the file `OEBPS_HOME\ebmsapps\<Application_Name>\maps\<WS_Name>\config.xml` file, where **WS_Name** is the name of the workstep.

**Contents**

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>
```

**Schema**

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: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 3: 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>
<tr>
<td>Element</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</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|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. |

**mapping.prop**

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

The format of this file is shown below.

```
adapter-configurator=<adapter configurator class name [optional]>
map-configurator=<map configurator class name [optional]>
runtime-mapper=<runtime mapper class name>
AdapterCategory=<adapter category [optional]>
AdapterName=<adapter name [optional]>
adapter-class=<adapter class name>
log-level=<log level [optional]>
adaplet=[true/false]
```

Example

An example of this file is provided below.

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

input.map and output.map

These two files are used to store the input and output maps for an adapter.

Location

The files are located in the `<Process_Directory>\maps\<WS_Name>` directory.

Important note

The information in this section is given as a reference only. Neither Application Users nor Adapter Developers are expected to directly manipulate the mapping files.

Usage notes

Adapters introduced after Business Process Server 6.0 store the input and output map files in XML. Business Process Server versions prior to 6.0 stored this mapping data as a property file. Typically, the new map configurator (`com.savvion.sbm.adapters.framework.GroupMapConfigurator`) generates XML files, while the simpler one-tab map configurator (`com.savvion.sbm.adapters.framework.TwoWayMapConfigurator`) generates property files. Because both formats are supported by the run-time mapper, you can use either of the map configurators for custom adapters.

Property Map format

The format generated by the simpler one-tab map configurator is presented below (for the XML map format, see XML Map format on page 27):

- Input Map
  
  `AdapterParameterName=java_type:value...`
• Input Map (XML Format)

```
<map version="1.1">
<param name="<AdapterParameterName>" type="
<java_type:value>" value="" OR @<DSNAME> />
...</map>
```

• Output Map

```
DataslotName=java_type:value...
```

• Output Map (XML Format)

```
<map version="1.1">
<param name="<AdapterParameterName>" type="
<java_type:value>" value="" OR @<DSNAME> />
...
</map>
```

**Adapter maps**

The maps are explained in the following table.

**Table 4: Adapter Maps**

<table>
<thead>
<tr>
<th>Map parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdapterParameterName</td>
<td>The name of an adapter parameter, whose value is set by the mapper. The adapter parameter should be of type &quot;I&quot;, &quot;IO&quot;, or &quot;C&quot;.</td>
</tr>
<tr>
<td>DataslotName</td>
<td>The name of a dataslot, whose value is set by the mapper. The dataslot should be output or input/output for the workstep.</td>
</tr>
<tr>
<td>java_type</td>
<td>The Java data type, to which the data should be converted before setting the adapter parameter.</td>
</tr>
<tr>
<td>value</td>
<td>The value received from the dataslot or the output value as transferred to the dataslot.</td>
</tr>
</tbody>
</table>

**ConfigForm example**

The following example of the *ConfigForm* generates the input/output maps as shown in *Figure 2: Interaction between components of the Managed Adapter framework* on page 21.

```
<configform synchronous="true" version="1.3">
  <block title="Adapter Data" label="Configure input and output Excel cells" hidden="false">
    <param name="A10" access="I" label="A10" type="java.lang.Object" value=""/>
    <param name="Sheet2_B3" access="O" label="Sheet2_B3" type="java.lang.Object" value=""/>
  </block>
  <block title="Configuration" label="Specify the workbook and default worksheet" hidden="false">
    <param name="WORKBOOK" access="PRESET" label="Workbook" type="java.lang.String" value="D:\new1.xls"/>
    <param name="WORKSHEET" access="PRESET" label="Default worksheet" type="java.lang.String" value="Sheet1"/>
    <param name="WORK_DIRECTORY" access="PRESET" label="Working OpenEdge Business Process Server: Managed Adapters Guide 26" value=""/>
  </block>
</configform>
```
Input mapping
The input datasets mapping is as follows.

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
The output mapping from the adapter is provided below.

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

```
<map version="1.1">
<param name="INPUTS" type="java.lang.String" value="A10"/>
<param name="OUTPUTS" type="java.lang.String" value="Sheet2!B3"/>
<param name="WORKSHEET" type="java.lang.String" value="Sheet1"/>
<param name="WORKBOOK" type="java.lang.String" value="D:\\new1.xls"/>
<param name="A10" type="java.lang.Object" value="@dsin"/>
<param name="WORK_DIRECTORY" type="java.lang.String" value="D:\temp"/></map>
```

• Output Map

```
<map version="1.1">
<param name="dsout" type="java.lang.String" value="@Sheet2_B3"/>
</map>
```

Supported data types
The Managed Adapter Configuration and Mapping Framework supports two-way conversion between various Target types and Source types, as listed in the following table.
## Supported data types for conversion

### Table 5: Supported data types for conversion

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

S: supported, no conversion performed  
SF: supported, if the source is in proper format  
SC: supported, conversion performed  
NS: not supported

### Rules

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

- **Converting numeric values.** Some care must be exercised when 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.

### OpenEdge BPM dataslots to Java data types conversion

The Managed Adapter dataslot configuration supports OpenEdge BPM dataslots to Java data types conversion, as listed in the following table.

#### OpenEdge BPM dataslots to Java data types conversion

**Table 6: Supported data types for conversion**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
<td>S</td>
<td>SF</td>
<td>SF</td>
<td>SF</td>
<td>SF</td>
<td>SF</td>
<td>SF</td>
<td>SF</td>
<td>S</td>
<td>NS</td>
</tr>
<tr>
<td>Integer</td>
<td>SC</td>
<td>SC</td>
<td>S</td>
<td>SC</td>
<td>SC</td>
<td>SC</td>
<td>SC</td>
<td>SC</td>
<td>S</td>
<td>NS</td>
</tr>
<tr>
<td>Int64</td>
<td>SC</td>
<td>S</td>
<td>SC</td>
<td>SC</td>
<td>SC</td>
<td>SC</td>
<td>SC</td>
<td>SC</td>
<td>S</td>
<td>NS</td>
</tr>
<tr>
<td>Decimal</td>
<td>SC</td>
<td>SC</td>
<td>SC</td>
<td>SC</td>
<td>SC</td>
<td>S</td>
<td>SC</td>
<td>SC</td>
<td>S</td>
<td>NS</td>
</tr>
<tr>
<td>Double</td>
<td>SC</td>
<td>SC</td>
<td>SC</td>
<td>S</td>
<td>SC</td>
<td>SC</td>
<td>SC</td>
<td>SC</td>
<td>S</td>
<td>NS</td>
</tr>
<tr>
<td>Logical</td>
<td>SC</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>S</td>
<td>NS</td>
</tr>
<tr>
<td>Object</td>
<td>SC</td>
<td>SF</td>
<td>SF</td>
<td>SF</td>
<td>S</td>
<td>SF</td>
<td>SF</td>
<td>SF</td>
<td>S</td>
<td>NS</td>
</tr>
<tr>
<td>DateTimeTZ</td>
<td>SC</td>
<td>SC</td>
<td>NS</td>
<td>SC</td>
<td>SF</td>
<td>SC</td>
<td>SC</td>
<td>NS</td>
<td>S</td>
<td>NS</td>
</tr>
<tr>
<td>Document</td>
<td>SC</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>S</td>
</tr>
<tr>
<td>Raw</td>
<td>SC</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>S</td>
</tr>
<tr>
<td>LongChar</td>
<td>S</td>
<td>SF</td>
<td>SF</td>
<td>SF</td>
<td>SF</td>
<td>SF</td>
<td>SF</td>
<td>SF</td>
<td>SF</td>
<td>NS</td>
</tr>
</tbody>
</table>

S: supported, no conversion performed  
SF: supported, if the source is in proper format (True or False)  
SC: supported, conversion performed  
NS: not supported
Caution: The usage of the LongChar dataslot is suggested only if the data is larger than 32k. For any smaller value, the usage of Character dataslot is suggested. Using Longchar dataslot might cause significant increase in response time.

The maxPostSize value for LongChar feature is set to 2 MB in the DatabaseMapping file in the location: $DLC$\ebpm\server\conf\resources\common. To increase the size to 5MB, edit the &lt;mapping&gt; section as shown below:

```xml
<mapping>
  <java>com.progress.lang.Longchar</java>
  <dataslot>ABL_LONGCHAR</dataslot>
  <database>LVARCHAR</database>
  <database-mb>LVARCHAR</database-mb>
  <size bytes="2097152">2097152</size>
</mapping>

to

<mapping>
  <java>com.progress.lang.Longchar</java>
  <dataslot>ABL_LONGCHAR</dataslot>
  <database>LVARCHAR</database>
  <database-mb>LVARCHAR</database-mb>
  <size bytes="5242880">5242880</size>
</mapping>

Also, edit the server.xml file in the location: $DLC$\ebpm\jboss\server\portalServer\deploy\jbossweb.sar to add maxPostSize attribute as shown in the following tables:

```xml
<Connector protocol="HTTP/1.1" port="18793" address="${jboss.bind.address}" connectionTimeout="20000" redirectPort="18002" URIEncoding="UTF-8" />

to

<Connector protocol="HTTP/1.1" port="18793" address="${jboss.bind.address}" maxPostSize="5242880" connectionTimeout="20000" redirectPort="18002" URIEncoding="UTF-8" />
```

Java data types to OpenEdge dataslots conversion

The Managed Adapter dataslot configuration supports Java data types to OpenEdge dataslots conversion, as listed in the following table.

Table 7: Supported data types for conversion

<table>
<thead>
<tr>
<th>Source data type</th>
<th>Character</th>
<th>Integer</th>
<th>Int64</th>
<th>Decimal</th>
<th>Double</th>
<th>Logical</th>
<th>Object</th>
<th>Date</th>
<th>TimeTZ</th>
<th>Decinet</th>
<th>Raw</th>
<th>LongChar</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.lang.String</td>
<td>S</td>
<td>SF</td>
<td>SF</td>
<td>SF</td>
<td>SF</td>
<td>SC</td>
<td>S</td>
<td>SF</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>S</td>
</tr>
<tr>
<td>java.lang.Long</td>
<td>SC</td>
<td>SC</td>
<td>S</td>
<td>SC</td>
<td>SC</td>
<td>NS</td>
<td>S</td>
<td>SC</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>SC</td>
</tr>
<tr>
<td>Source data type</td>
<td>Character</td>
<td>Integer</td>
<td>Int64</td>
<td>Decimal</td>
<td>Double</td>
<td>Logical</td>
<td>Object</td>
<td>Date TimeTZ</td>
<td>Dmamt</td>
<td>Raw</td>
<td>LongChar</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>-----------</td>
<td>---------</td>
<td>-------</td>
<td>---------</td>
<td>--------</td>
<td>---------</td>
<td>--------</td>
<td>------------</td>
<td>-------</td>
<td>-----</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>java.lang. Integer</td>
<td>SC</td>
<td>S</td>
<td>SC</td>
<td>SC</td>
<td>SC</td>
<td>NS</td>
<td>S</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td></td>
</tr>
<tr>
<td>java.lang. Double</td>
<td>SC</td>
<td>SC</td>
<td>SC</td>
<td>SC</td>
<td>S</td>
<td>NS</td>
<td>S</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td></td>
</tr>
<tr>
<td>java.lang. Float</td>
<td>SC</td>
<td>SC</td>
<td>SC</td>
<td>S</td>
<td>SC</td>
<td>NS</td>
<td>S</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td></td>
</tr>
<tr>
<td>java.lang. BigDecimal</td>
<td>SC</td>
<td>SC</td>
<td>SC</td>
<td>SC</td>
<td>SC</td>
<td>NS</td>
<td>S</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td></td>
</tr>
<tr>
<td>java.sql. Timestamp</td>
<td>SC</td>
<td>NS</td>
<td>SC</td>
<td>SC</td>
<td>SC</td>
<td>NS</td>
<td>S</td>
<td>SC</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td></td>
</tr>
<tr>
<td>java.lang. Boolean</td>
<td>SC</td>
<td>SC</td>
<td>SC</td>
<td>SC</td>
<td>SC</td>
<td>S</td>
<td>S</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td></td>
</tr>
<tr>
<td>java.lang. Object</td>
<td>SC</td>
<td>SF</td>
<td>SF</td>
<td>SF</td>
<td>SF</td>
<td>SF</td>
<td>S</td>
<td>SC</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td></td>
</tr>
<tr>
<td>java.lang. Byte[]</td>
<td>SC</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>S</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
<td></td>
</tr>
</tbody>
</table>

S: supported, no conversion performed
SF: supported, if the source is in proper format (True or False)
SC: supported, conversion performed
NS: not supported
**Caution:** The usage of the LongChar dataslot is suggested only if the data is larger than 32k. For any smaller value, the usage of Character dataslot is suggested. Using Longchar dataslot might cause significant increase in response time.

The `maxPostSize` value for LongChar feature is set to 2 MB in the DatabaseMapping file in the location: `$DLC$\oebpm\server\conf\resources\common`. To increase the size to 5MB, edit the `<mapping>` section as shown below:

```xml
<mapping>
<java>com.progress.lang.Longchar</java>
<dataslot>ABL_LONGCHAR</dataslot>
<database>LVARCHAR</database>
<database-mb>LVARCHAR</database-mb>
<size bytes="2097152">2097152</size>
</mapping>
```

to

```xml
<mapping>
<java>com.progress.lang.Longchar</java>
<dataslot>ABL_LONGCHAR</dataslot>
<database>LVARCHAR</database>
<database-mb>LVARCHAR</database-mb>
<size bytes="5242880">5242880</size>
</mapping>
```

Also, edit the `server.xml` file in the location: `$DLC$\oebpm\jboss\server\portalServer\deploy\jbossweb.sar` to add `maxPostSize` attribute as shown in the following tables:

```xml
<Connector protocol="HTTP/1.1" port="18793" address="${jboss.bind.address}" connectionTimeout="20000" redirectPort="18002" URIEncoding="UTF-8" />
```

to

```xml
<Connector protocol="HTTP/1.1" port="18793" address="${jboss.bind.address}" maxPostSize="5242880" connectionTimeout="20000" redirectPort="18002" URIEncoding="UTF-8" />
```

**Supported APIs**

This section describes the various APIs supported by the Managed Adapter Framework and presents a sample code for them.

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

public interface MapConfigurator {
    public void buildMap(Hashtable context) throws Exception;
    public void buildMap(Hashtable context, Hashtable inputDataslots, Hashtable outputDataslots) 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;
}

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.

```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;
    }
}
```

Adapter logging

Adapter logging is controlled by the `logLevel` parameter in the `mapping.prop` file (see `mapping.prop` on page 24). The `logLevel` is a non-negative integer, and a higher log level indicates more detailed logging. The default `logLevel` for most Managed Adapters is 10.

A separate log file is created for each workstep that is performed by a Managed Adapter. Table 8: Log file location for applications on page 34 provides the location of these log files.

Table 8: Log file location for applications

<table>
<thead>
<tr>
<th>Application</th>
<th>Log file location</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP Server</td>
<td>OEBPS_HOME\ebmsapps\AppName\log\WSName.log</td>
</tr>
<tr>
<td>Web application</td>
<td>OEBPS_HOME\BPM Webflow\AppName\log\WSName.log</td>
</tr>
<tr>
<td>Private Webapps</td>
<td>OEBPS_HOME\ebmsapps\AppName\webapp\PrivateWebAppName\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.
Using the Adapter Configurator and Map Configurator

When a Managed Adapter is used in a process, you must define the mapping from input dataslots in your process to the adapter inputs in an external application and from the adapter outputs to your output dataslots. The MapConfigurator is the GUI-based component used to define this mapping.

Progress Developer Studio for OpenEdge (or Business Process Modeler) provide a default MapConfigurator that can be used for all out-of-the-box adapters. However, the MapConfigurator can also be defined separately for each adapter.

The MapConfigurator is started, when a Managed Adapter is used as the workstep performer.

In the following example, the Email Adapter is used to illustrate mapping. The following procedure assumes you have already created a preconfigured instance of this adapter.

1. Create a Managed Adapter workstep and assign Email Adapter as its performer. Double-click the workstep to open the workstep’s Properties dialog box.

2. From the Properties view of this workstep, open the Configuration tab.

You can specify other properties in the rest of tabs in the Properties view, same as that for custom adapters. For information regarding configuring properties for custom adapter workstep, refer to the "Defining Properties of Adapter Worksteps" section of the OpenEdge Getting Started: Developing BPM Applications with Developer Studio.

3. To configure the Managed Adapter, click Configure..., to open the AdapterConfigurator for the Email Managed Adapter.

   Note: You can also open the Email Adapter Configurator using the new Managed Adapter Browser (available only from Progress Developer Studio for OpenEdge) functionality. In addition to configuring, you can use the Managed Adapter Browser to perform functions like creating a copy, renaming, deleting, as well as importing and exporting the configuration information. For more information, refer to the "Using the Managed Adapter Browser" section of the OpenEdge Getting Started: Developing BPM Applications with Developer Studio.

   a) In the Adapter Configurator, you can enter new, or modify existing, information that are implemented only for the adapter instance. Open the Message tab and enter information in the From field and the Subject field. We leave the To field blank for now because it is mapped to a dataslot in Step 4.

   b) Open the Message Body tab and enter information. Note that we use the $ symbol to create variables that are mapped to dataslots. Click Extract Variables to display the variables in the text area below.

   c) Open the Configuration tab to configure the Mail Host and enter the notation "$\{oebps.smtp.host\}" to tell the adapter to read the Mail Host name from the oebps.smtp.host property in the oebps.conf configuration file.

   d) Enter the Username and Password details and select the language from the Language dropdown.

   e) When you have finished configuring the Email Adapter, click OK, to open the Dataslot Mapping dialog box (or the Map Configurator). You can also open the Map Configurator by clicking the Change mapping button.

4. To modify the dataslot mapping, click Change mapping from the Configuration tab of the workstep’s Properties view, to open the Dataslot Mapping dialog box of the map configurator for the adapter. This example demonstrates how to send a message to all employees on a mailing list stored in a property file.

   Note: The dataslot mapping dialog box are different for each adapter and each adapter configuration.
a) In the **Message** tab of the **Dataslot Mapping** dialog box, you can define the mapping between the input/output dataslots and the adapter inputs/outputs by either:

- Selecting one from the drop-down list.
- Using the "GroupMapper" feature by entering a notation that refers to email addresses, such as an employee mailing list, that are stored in a property file. This notation instructs the adapter to get the "empMailingList" property from the "mail.prop" file and use the property value as the email destination.

b) Open the **Velocity Template** tab to map dataslots to each of the variables we used in the message body. Map the "meetingRoom" dataslot to the "roomNumber" variable to substitute the value of this dataslot for the $roomNumber field in the message text; and similarly map the "schedTime" dataslot to the "time" variable.

c) Click **OK** to return to the **Configuration** tab of the workstep’s **Properties** view.

5. To edit the **mapping.prop** file to change the settings for mapping (for example, log level), click **Advanced properties** from the **Configuration** tab of the workstep’s **Properties** view, to open the **Advanced Adapter Properties** dialog box.

a) Edit the **mapping.prop** file displayed in the **Advanced Adapter Properties** dialog box appropriately as per your requirement.

b) Click **OK** to return to the **Configuration** tab of the workstep’s **Properties** view.

c) Click **OK** to save the changes.

The **Dataslot Mapping** dialog box lists all input and output dataslots for the workstep, and lists all the visible adapter parameters, grouped according to the **config.xml**. You can enter or select the value of the given dataslot or parameter. Simple syntax distinguishes constant values from values to be copied from dataslots or from adapter parameters. The following table explains the syntax in detail.

**Table 9: Mapping syntax**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@&lt;dataslot&gt;</td>
<td>The value of the corresponding adapter input comes from:</td>
</tr>
<tr>
<td></td>
<td>• The given dataslot. OR</td>
</tr>
<tr>
<td></td>
<td>• The value of the output dataslot is transferred from the given adapter output.</td>
</tr>
</tbody>
</table>

Usually you select the dataslot from the drop-down list, making it unnecessary to manually enter the @ symbol. Once a valid dataslot is selected, the @ symbol is not displayed, but an icon representing the dataslot type is shown.
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Begins with any other character</td>
<td>Any character other than &quot;@&quot; at the beginning of a string signals that the value is a constant. In case the constant itself begins with &quot;@&quot;, precede the &quot;@&quot; with &quot;&quot;; or if the constant begins with a &quot;&quot;, use &quot;&quot;. The &quot;&quot; and &quot;@&quot; symbols are significant only at the beginning of the string, and have no special meaning anywhere else in the string.</td>
</tr>
<tr>
<td><code>${propertyFile: propertyName}</code></td>
<td>If an adapter input is mapped to a constant (see the previous row in this table), you can use this &quot;GroupMapper&quot; notation to extract values from property files on the server. The &quot;propertyFile:&quot; prefix is optional — if no property file is defined, &quot;oebps.conf&quot; are read. For example, &quot;$oebps.home&quot; is substituted with the value of the &quot;oebps.home&quot; property from the &quot;oebps.conf&quot; file. If you use your own property files, make sure they are located in the server classpath (EJB Server or Portal Server, depending on where your application is running). Custom property files may contain references to another file by including &quot;alias=fileName&quot; property. For example, if &quot;file1.prop&quot; includes &quot;alias=file2.prop&quot;, the property values from &quot;file2.prop&quot; are read too. In case a property is present in both files, higher precedence are given to the value in the &quot;file2.prop&quot; file.</td>
</tr>
</tbody>
</table>

**Note:** Even when the input and output have exactly same format, copying a value from an input dataslot directly to an output dataslot is not allowed.

Note that some of the adapter parameters have read-only combo boxes when:

- The parameter is output-only.
- The parameter is a constant.

When map configuration is completed, the MapConfigurator creates the input and output map files in the `OpenEdge\workspace directory under <Process_Name>\maps\<Workstep_Name>`.

### Deploying maps using MapDeployer

The deployment of the maps is the process of transferring the map files and the associated files to a database repository, where they will be accessible to the RuntimeMapper.

After you complete the mapping, the process is automatically published from Progress Developer Studio for OpenEdge.

However, if needed, you can install the process manually by deploying the maps separately.

**To deploy the maps, use the following command:**

```
MapDeployer processTemplateName [ -t processType ] [ -d processTemplateDirectory ]
```

The `[-t processType]` optional parameter is used to define the type of process in which the adapter will be used. If the adapter will be used in a Web application (formerly a BPM Webflow application), specify `[-t BPM Webflow]`. Otherwise, you need not specify the parameter.

The `[-d processDirectory]` optional parameter is used to specify a process directory, if the directory is non-standard. The standard process directories are `OEBPS_HOME\ebmsapps\ProcessName` or `OEBPS_HOME\BPM Webflow\ProcessName`. 

---

*OpenEdge Business Process Server: Managed Adapters Guide*
Troubleshooting map deployment

The MapTool, a command-line tool for manipulating the repository, can check if a map is deployed properly and enables you to perform other operations on the database map repository. If you invoke MapTool.{sh|bat} with no command-line arguments, it displays the following list of available options and commands:

Usage: MapTool { context } { commands }

Context is specified as a set of following options:
- `–pt ###` - specify process template name
- `–pwa ###` - specify private webapp name
- `–ws ###` - specify workstep name
- `–t ###` - specify process type
- `–d ###` - specify process template directory
- `–clear` - clears the context

Valid commands are:
- `deploy` - deploy a map
- `deployAll` - deploy all maps in a process
- `undeploy` - undeploy a map
- `undeployAll` - undeploy all maps in a process
- `isDeployed` - check if the given map is deployed
- `show` - show a content of a deployed map
- `showLocal` - show a content of a local (not deployed) map

The argument list may seem complex at first sight, but actually MapTool is quite simple to use. You can manipulate maps using any of the commands listed under the "Valid commands" section. Commands are executed in the order they are given. Before a command is given though, you must state the appropriate "context", uniquely identifying the map or maps the command should operate on.

For example, consider that you want to use MapTool to show the content of a deployed map for a workstep named "WS1" of a process called "MyProcess". The command to show the content of the already deployed map would be "show", as listed in the MapTool help message. To uniquely identify the map by process and workstep name, use the following command:

MapTool –pt MyProcess –ws WS1 show

The first four parameters, "–pt MyProcess –ws WS1", set the context required for the command that follows: "show".

You can issue a number of commands at the same time, each one preceded by a change in the context. For example, if you want to see not only the map for the workstep "WS1", but also the map for another workstep--"WS2",--you can combine both commands as follows:

MapTool –pt MyProcess –ws WS1 show –ws WS2 show

As previously stated, the MapTool processes the arguments one by one, in the order they are given on the command line. The "–pt MyProcess –ws WS1" options set the context for the first "show" command. After the "show" command is executed, MapTool continues processing the rest of the arguments. The next two arguments, "–ws WS2", change the workstep name, while the rest of the context (the process name) is preserved, thus providing for the execution of the second "show" command.

You can may also execute several commands in the same context, as shown in the following example:

MapTool –pt MyProcess –ws WS1 deploy show

In this case, you instruct MapTool to deploy a map, and immediately after this to verify that it has been properly deployed.
Database Managed Adapter

The Database Adapter is one of the Managed Adapters conforming to the Adapter Configuration and Mapping Framework. For more information on this framework, refer to Understanding the Managed Adapter framework on page 19.

The Database Adapter provides:

• A Configuration GUI to specify database connections, SQL statements, variables, and parameters.

• A stand-alone mapping tool, as well as one embedded within Progress Developer Studio for OpenEdge to define mapping between dataslots and database values.

• The Database Adapter class, which connects to the specified databases, executes the specified SQL statements, and calls the Run-time Mapper to map dataslots and database values.

The Database Adapter supports the following databases:

• OpenEdge

• DataDirect Cloud

• Oracle

• Microsoft SQL Server

The Database Adapter helps application developers to quickly develop applications that access databases without additional programming efforts.

For details, see the following topics:

• Database Adapter features

• Database Adapter components
Database Adapter features

Business Process Server applications need to access a database to retrieve and save information. For this purpose, Application Developers design an adapter workstep and use JDBC programming in the adapter class. However, writing JDBC code from scratch is required every time, which can be streamlined using the Database Adapter.

The Database Adapter includes the following features:

- An easy-to-use adapter which is seamlessly integrated with the Progress Developer Studio for OpenEdge.
- A pre-built adapter class that loads information dynamically from XML and carries out the execution and mapping automatically, without any additional coding.
- Sequential execution of multiple SQL statements and connecting to multiple databases.
- The DB Adapter uses the default MapConfigurator GUI as described in Understanding the Managed Adapter framework on page 19

Database Adapter components

The following are the major components of the Database Adapter:

- **Database Adapter Configurator**: Swing-based GUI to configure database connection parameters, SQL statements, and variables that would be mapped to input values and query results.
- **Adapter Configuration File**: AdapterConfigurator saves relevant information into this XML file.
- **MappingConfigurator**: MappingConfigurator defines the mapping between dataslots and adapter variables.
- **Adapter Mapping Files**: MappingConfigurator saves mapping information into an input mapping XML file and an output mapping XML file.
- **Run-timeMapper**: A pre-built class which reads Adapter Mapping Files and maps the values at run time.
- **Database Adapter class**: A pre-built class which handles query execution. The Database Adapter worksteps use it as the performer.
**Working with the Database Adapter**

The following section describes how to configure the Database Adapter.

**Configuring the Database Adapter**

The Database Adapter needs three types of information:

- Database connection parameters
- SQL statements
- Input and output variables of the adapter

The Database Adapter Configurator GUI is designed to get the above information, and save it in the Adapter Configuration File.

**Defining variables for the Database Adapter**

To define variables:
1. After assigning the DBAdapter Managed Adapter as performer of a workstep, click the workstep to open its Properties view.

2. Open the Configuration tab and click the Configure button to display the Database Adapter Configurator GUI (You can alternatively use the DBAdapter.cmd/.sh script from the OEBPS_HOME/bin folder to invoke the Database Adapter Configurator GUI). The DBAdapter Configurator dialog box is displayed as shown in the following figure.

   **Note:** You can also open the DBAdapter Configurator using the Managed Adapter Browser dialog box (available only from Progress Developer Studio for OpenEdge) functionality from the Tools > Managed Adapters menu. In addition to configuring, you can use the Managed Adapter Browser to perform functions like creating a copy, renaming, deleting, as well as importing and exporting the configuration information. For more information, refer to the "Using the Managed Adapter Browser" section of the OpenEdge Getting Started: Developing BPM Applications with Developer Studio.

   If this is a new process, the SQL Statements List table is empty. If there are existing statements present, they are listed in the order they are to be executed. You can use Move Up and Move Down buttons to change the execution order.

3. Click Variables to open the Variable Definitions dialog box. You can Add, Modify, and Remove variable definitions from this window.

4. Click Add to open the Variable Editor dialog box. You can also select a variable from the Variable Definition dialog box and click Modify to open the same box.
   a) Enter the Variable Name.
   b) Select the Type and Access from the drop-down list.
      - The supported Java data types are String, Integer, Long, Boolean, Date, Object, and List.
      - The supported access types are Input, Output, Input/Output, and Local.
   c) You can optionally provide a Default value and Description for the variable. For Date datatype, the supported default value formats are "dd/mm/yy", "mm/dd/yy", "dd/mm/yyyy hh:mm", and "mm/dd/yyyy hh:mm." For Boolean datatype, if the default value is empty, it is considered as "false." Any other value, except "true" (non case-sensitive), is also considered as "false."
   d) Click OK to return to the Variables Definitions dialog box. The added variable is now displayed in the variable list.

5. Click OK to return to the DBAdapter Configurator dialog box.

**Defining Database connection**

You can use the DBAdapter Configurator dialog box to add, modify, and remove SQL statements. While defining or editing SQL statements, you can specify the database connection details.

**To define database connection:**

1. To create an SQL statement, click Add to open the SQL Statement dialog box. To modify an existing SQL statement, select an SQL statement and click Modify.

   The SQL Statement dialog box provides four tabs: Connection, SQL Statement, Input Parameters, and Output Parameters. Initially, only the Connection tab is enabled.

2. Enter the SQL Statement Name.
3. To define Database Parameters, click **Choose a Database** to open the **Database Definitions** dialog box that lists available databases. You can add, modify, and remove Database definitions.

4. To define a database definition, click **Add** to open the **Database Parameters** dialog box. To modify an existing database definition, select a Database name and click **Modify**.
   a) Enter a name in the Database Name box.
   b) Select a database type from the Database Type drop-down list. Supported Database types include OpenEdge.
   c) Enter the Login User name and Password.
   d) Select the Database type to populate data in the JNDI Name field and in the Driver and URL fields in the Non-J2EE Datasource parameters panel. You need to modify this data with the proper database information.
   e) Click **OK** to return to the **Database Definitions** dialog box. The added database definition now displays in the definition list.

5. Select a database and click **OK** to return to the **SQL Statement** dialog box.

6. Click **Connect** to connect to the selected database.
   The Status panel displays whether the connection is successful. If it is successful, the message “Successfully connected to database: <database_name>” appears in the Status panel. The **SQL Statement** tab is also enabled so you can specify the SQL statement.

7. Click the **SQL Statement** tab to define the SQL statements.

**Defining SQL statements**

In the SQL Statement tab, you can construct an SQL statement.

---

**Note:** For MS SQL Server, ensure that you make use of alias names for database functions in the SQL query to avoid runtime exception and suspension of the adapter workstep. For example, `select UPPER(USER_NAME) AS USERNAME from UMUSER where USERNAME is the alias name for the database function, UPPER(USER_NAME)`.

---

**To define SQL statements:**

1. Choose the SQL Statement type from the following options: **Select**, **Insert**, **Update**, **Delete**, or **Stored Procedure**. These options are used to define an SQL Statement, and are described below:
   - **Select.** Enables you to select an existing record (or row) from a table to define a new SQL Statement.
   - **Insert.** Allows you to insert a record into a table.
   - **Update.** Enables you to update one or more selected records in a table.
   - **Delete.** Enables you to delete one or more selected records in a table.
   - **Stored Procedure.** Enables you to choose a procedure from ones you have defined previously and stored, and use it as a new SQL Statement.

2. Depending on the type you choose, one or more corresponding buttons are displayed in the Clauses pane.

3. Click **Choose Tables**. The list of Available Tables is displayed in the **Choose Table(s)** dialog box.
   a) Choose the required tables. Based on the selection and the statement type, the configuration tool fills in one of the following into the statement:
• Select from [tables]
• Insert into [table]
• Update [table]
• Delete from [tables]

b) Click **OK** to return to the **SQL Statement** tab.

4. When your SQL statement type is **Select** and you have chosen a table, the **Choose Columns** button is enabled in the Clauses pane. Click **Choose Columns** to display the list of available columns for the selected table.

a) Choose the required columns for the syntax:

    Select [columns] from [tables]

b) Click **OK** to return to the **SQL Statement** tab.

5. When your SQL statement type is **Insert or Update** and you select a Table, the **Set Values** button is enabled in the Clauses pane. Click **Set Values** to open the **Select Columns & Set Their Values** dialog box, which presents a list of Available Column.

a) Choose the required columns, enter its Value directly, or click in the Use Variable column to open the **Variable Definitions** dialog box, where you can take a value from an input variable. The configuration tool fills in the following syntax:

    Insert into [table] (columns) (values)
    Update [table] set [column1=value1, ...]

b) Click **OK** to return to the **SQL Statement** tab.

   The input parameters for the SQL statements are usually provided through input or internal variables. The column value, however, may be a constant or another SQL statement.

6. When your SQL statement type is **Select, Update, or Delete**, the **Add Conditions** button is available in the Clauses pane. Click **Add Conditions**, to open the **Compose a Condition Clause** dialog box.

   The condition clause is composed of expressions in the "column-operator-value" format.

a) Select a column from the Database Columns drop-down list.

b) Click **Append a Column** to append it to the condition clause.

c) Select an operator from the SQL Operators drop-down list.

d) Click **Append an Operator** to append it to the condition clause.

e) When you need to use an input variable, click **Append a Variable** to open the **Variable Definitions** dialog box, where you can select and append one of the listed variables. For more information on adding variables, see Step 3.

   The SQL statement is displayed in the SQL Statement Preview pane.

f) Click **OK** to return to the **SQL Statement** tab.
7. If the above controls are not adequate for your SQL statement, you can select the **Expert Mode** button to directly modify the where clause. In this case, you need to be careful with the syntax, as SQL syntax parsing and verification is not carried out in the Expert mode. For more information, see Using the Expert mode on page 48.

8. When your SQL statement type is **Select**, you can click **Get Result MetaData** to open the **Query Tester** dialog box where you can verify your statements.

   The upper Input Sample Values pane displays the parameters embedded in the SQL statement and the input variables to which they map.

   a) Enter actual values for parameters in the Value column. Alternately, select the **Use Default Values** check box to take the parameter values from input variable default values.

   b) Click **Execute** to test the query. The Query Result pane shows the test results.

   c) Click **OK** to return to the SQL Statement tab.

   **Note:** When comparing 'char' database columns to variables, make sure that one of the following is true: 1) If an 'equal to' comparison is done and if the size in the variable is less than the size of the 'char' column, pad the variable value with spaces until the size of the 'char' column is matched. Only then does the 'equal to' comparison work. 2) Use a database-specific trim function to trim the value selected from the 'char' column before running the comparison with a variable value. This results in trimming down additional spaces from the column value before comparison.

---

**Defining input and output variables**

After successfully testing the Select statement, the **Output Parameters** tab is enabled to allow you to map values from ResultSet to output variables.

**To define input and output variables:**

1. Open the **Input Parameters** tab to display parameters embedded in the SQL statement, and the input variables to which they map.

2. Click the **Output Parameters** tab to display the output parameters.

   A database query may return one single row or multiple rows.

3. Select the option **Expect Single Row** if you expect one single row. Select the option **Expect Columns as List** if you expect a list of values that you map to a List variable. Select the option **Expect Multiple Rows** if you expect multiple rows.

   For multiple rows, you can only map the whole **ResultSet** to an Object type output variable. The Object dataslot in your process must be of type, com.sun.rowset.CachedRowSetImpl. For information regarding creating an Object dataslot, see Chapter 12, "Using Dataslots" in OpenEdge Getting Started: Developing BPM Applications with Developer Studio. The CachedRowSetImpl object can be processed by a custom adapter, or can be displayed as a table as described in the Displaying multiple rows selection section.

   For a single row, you can specify the output variable to which a column maps.

4. Click **OK** to complete the Database Adapter configuration.

   To configure the installed Database Adapter, use the **DBAdapter.cmd|sh script** and save the adapter configuration in the OEBPS_HOME\managedadapters\DB\DBI directory with the name config.xml.
Displaying multiple rows selection

The CachedRowSet object, returned by the adapter when the "Expect Multiple Rows" option is checked, can be displayed on a Web application or BP Server JSP page by using the displayResultSet tag. The dataslot to be displayed should be configured as an Input or Input/Output dataslot for that workstep.

You can edit the JSP page, replacing the following tag

```html
<bizsolo:getDS name="DS_NAME"></bizsolo:getDS>
```

with

```html
<bizsolo:displayResultSet name="DS_NAME"/>
```

to display the table.

**Note:** You can edit only custom JSPs to replace the tag for CachedRowSet object.

Here, DS_NAME is the name of the dataslot containing the CachedRowSet object. The tag generates HTML presentation for the dataslot.

The default look of the table can be modified, by specifying additional attributes to the displayResultSet tag. The full list of attributes is shown below:

```html
<bizsolo:displayResultSet
name = "DS_NAME"
tableOptions = "TABLE_OPTIONS"
headerOptions = "HEADER_OPTIONS"
bodyOptions = "BODY_OPTIONS"
altBodyOptions = "ALT_BODY_OPTIONS"
headerFontOptions = "HFOPTS"
bodyFontOptions = "BFOPTS"
headerTags = "HT1,HT2"
bodyTags = "BT1,BT2"
columnOptions = "COL_OPT1,COL_OPT3"
columnNames = "COL1,,COL3"
startFrom = "START_FROM"
maxRows = "MAX_ROWS"
/>
```

Most of these tags are directly related to the HTML that are generated. The attribute values are directly inserted in specific places of the HTML, as shown below:

```html
<TABLE TABLE_OPTIONS>
<TR HEADER_OPTIONS>
    <TD COL_OPT1><HT1><HT2><FONT HFOPTS>COL1</FONT></HT2></HT1>
    <TD></TD>
</TR>
<TR BODY_OPTIONS>
    <TD COL_OPT1><BT1><BT2><FONT BFOPTS>data</FONT></BT2></BT1>
    <TD></TD>
</TR>
<TR ALT_BODY_OPTIONS>
    <TD COL_OPT1><BT1><BT2><FONT BFOPTS>data</FONT></BT2></BT1>
    <TD></TD>
</TR>
</TABLE>
```
Table 10: Attributes used in configuring the Database Adapter on page 47 describes the attributes used.

### Table 10: Attributes used in configuring the Database Adapter

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Specify the name of the dataslot.</td>
</tr>
<tr>
<td>columnNames</td>
<td>By default, table header is generated by the names of the database table fields. The columnNames attribute includes a comma-separated list of alternative (human-readable) values, to be used in the table header. Some of the column names can be left empty-this means that the corresponding column is not displayed. For example, assuming that a DB query returns a result set with three fields, COL1, COL2, and COL3, setting the columnNames attribute to columnNames = &quot;Column 1,, Column 3&quot; display only the first and third column, and changes the names displayed in the header from COL1 and COL3 to Column 1 and Column 3.</td>
</tr>
<tr>
<td>startFrom maxRows</td>
<td>In case the result set contains a large number of records, these two tags allow you to display only a portion of this result set. With the startFrom attribute you can set the first row to be displayed (starting from 1), while the maxRows attribute limits the maximum number or rows that the table contains. All rows, outside of the section defined with the startFrom and maxRows attributes, are not displayed.</td>
</tr>
<tr>
<td>headerTags bodyTags</td>
<td>Tags for controlling the appearance of the data in the cells of the table header and body. Each of these attributes contain a comma-separated list of tag names that are placed in pairs around each value in the header and body. For example, headerTags=&quot;B,I&quot; generates the following in each cell of the table header: <code>&lt;B&gt;&lt;I&gt;header data&lt;/I&gt;&lt;/B&gt;</code></td>
</tr>
<tr>
<td>tableOptions headerOptions bodyOptions altBodyOptions headerFontOptions bodyFontOptions columnOptions</td>
<td>Additional attributes for modifying the table style. Use standard HTML options for the values of each of these attributes. For example: headerOptions = &quot;bgcolor=red&quot; changes the background color of the table header to &quot;red&quot;. Consult the HTML specification for more information.</td>
</tr>
</tbody>
</table>

### Using stored procedures

You can execute a stored procedure that you have defined previously and stored, and use it as a new SQL Statement.

To use a stored procedure:

1. From the Configuration tab of the DBAdapter Properties dialog box, click Configure to open the DBAdapter Configurator dialog box.
   
   It displays the SQL Statements along with their types. You can add a new statement, or modify or delete selected statements. You can click Move Up and Move Down to change the order of the statements.

2. Click Add to open the SQL Statement dialog box.
3. Connect to a database and click the SQL Statement tab, which displays the available SQL statements.

4. Click Stored Procedure as the type, then click Choose Procedure to open the Stored Procedures dialog box, which displays the list of available stored procedures.

5. From the list of procedures in the Stored Procedures dialog box, select a procedure and click Procedure Info to display the details about the selected procedure.

6. In the DBAdapter Configurator dialog box, click Variables to open the Variable Definitions dialog box. Once a stored procedure is selected; the input, output and I/O variables for that procedure get created and added to the existing list of variables. The input and output parameters of the stored procedure also get mapped to the corresponding variables.

   You can add, modify or remove variables. If you change any variable, you must change the corresponding SQL statement appropriately.

7. From the SQL Statements dialog box, use the Input Parameters tab to view the defined input parameters.

8. Click the Output Parameters tab to view the defined output parameters.

   Note: Result of a stored procedure execution is always a single row. Hence, by default, Expect Single Row option is selected, and all other options are disabled.

9. Click OK to return to the DBAdapter Configurator dialog box. Click OK to return to the Properties view.

10. Click Change Mapping.

   For input parameters, only Source mapping can be defined. For output and return parameters only Target mapping can be defined. For I/O parameters, both Source as well as Target mapping can be defined. Note that this dialog box shows variables for all the stored procedures configured for a specific workstep.

Return values from stored procedures

This section explains various scenarios about return values from stored procedures.

• If a single cursor is returned from the stored procedure, then you need to map RETURN_CURSOR variable to an Object dataslot of class

   com.sun.rowset.CachedRowSetImpl

   The Database Adapter will check if the variable RETURN_CURSOR is mapped to an Object dataslot, and if it is mapped, then the Database Adapter will call getResultSet() on CallableStatement and will return the CachedRowSetImpl object to the mapped Object dataslot.

• If multiple cursors are returned from a stored procedure body, all of them can be retrieved from CallableStatement by calling getResultSet() and getMoreResults() methods in loop till getMoreResults() returns false. To retrieve all these ResultSets, you need to map the variable RETURN_CURSOR to a List dataslot.

   The DBAdapter will retrieve all ResultSets, convert them to CachedRowSetImpl objects, populate the List with these objects and will return it for the List dataslot.

Using the Expert mode

The Database Adapter Configurator provides the Expert mode, which enables you to enter SQL Statements directly.

To switch to Expert mode:
1. From the SQL Statement tab in the Database Adapter Configurator, click Expert mode to open the Confirm prompt.

2. Click Yes to leave the default Wizard mode and enter the Expert mode, and the following SQL Statement dialog box appears, with the Wizard Mode button now enabled.

3. You can manually modify the SQL Statement as required. You can still use variables for column values, but remember to precede the variable name with @. To leave the Expert mode, and return to the default Wizard mode, click the Wizard Mode button.

4. When you have completed entering the statement, click Variables in the Clauses panel to open the Variable Definitions dialog box, where you can automatically define all the variables specified in the statement (if those are not already defined). You must still specify the data type for each variable.

5. You must now define the variable type. Select the variable, click Modify and define the data type in the Variable Editor dialog box as required.

Using Dynamic SQL statements

There are situations when the SQL statement cannot be defined completely at configuration time. In Expert mode, select the Dynamic SQL checkbox to switch to dynamic SQL mode, where parts of the statement (or even the whole statement) are supplied by dataslot values.

Dynamic SQL can be useful in many situations. As an example, consider a case where customer information is stored in a simple database table, as shown below:

<table>
<thead>
<tr>
<th>ID (INTEGER)</th>
<th>Name (CHARACTER)</th>
<th>Phone (CHARACTER)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>George Smith</td>
<td>(111) 555-1111</td>
</tr>
<tr>
<td>101</td>
<td>Mike Green</td>
<td>(222) 555-2222</td>
</tr>
<tr>
<td>102</td>
<td>Jeff Black</td>
<td>(333) 555-3333</td>
</tr>
<tr>
<td>103</td>
<td>Dennis Gray</td>
<td>(444) 555-4444</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Table 11: Dynamic SQL statement example on page 49 provides a list of customer ID-numbers; the task is to use the DB Adapter to select the data about all the customers in the list.

Note: If you would like to try the Database Adapter with the example discussed here, instructions about creating the customer database table are given in Step 1 of Tutorial 2, in Using the Database Adapter in Progress OpenEdge Business Process Modeler: User's Guide.

The SQL statement for getting the customer data from an ID-list would be:

```
SELECT * FROM CUSTOMER_INFO WHERE ID IN (100, 103, 104)
```

In the above example, 100, 103, and 104 are the customer IDs to select. Since the number of IDs in this list can vary, we cannot use prepared SQL statements (where the number of parameters is fixed), and are forced to switch to dynamically constructing the SQL.

To create a dynamic SQL statement:
1. Open the **Properties** view of a DBAdapter workstep. Open the **Configuration** tab and click **Configure**, displaying the **Database Adapter Configurator** dialog box.

2. From the **SQL Statement** tab, click **Expert mode**. Select the **Dynamic SQL** checkbox. Click **Select** as the type. In the SQL Statement panel, enter the following:

   ```sql
   select * FROM CUSTOMER_INFO WHERE ID IN (@IDLIST)
   ```

   Note that we used the variable @IDLIST in place of the list of customer ID numbers.

3. Click **Variables** to specify the type for the @IDLIST variable.

   Although an individual Customer ID is a number, a comma-separated list of IDs is not—that is the reason why we must select "CHARACTER" as the variable type.

   When you have defined the @IDLIST variable, click **OK** to return to the **SQL Statement** dialog box.

4. Click **Get Result MetaData** in the **SQL Statement** dialog box to invoke the **Query Tester** dialog box. Enter a list of values for the @IDLIST variable, and click in the row or press **TAB** to enable the values. Click **Execute** to show the list of selected customers in the **QueryResult** panel.

Replacing the variable values in the SQL statement is literal—meaning that you should be very careful when dynamically generating statements. For example, if the list we were providing consisted of Strings, and not numbers, the SQL syntax would require that the values are enclosed in single-quotation marks:

```sql
'abc','def','xyz'
```

You must make sure that the string values you provide to the Database Adapter conform to the SQL syntax.

In the example above, we used a variable to substitute only a small part of the SQL statement — the list of ID numbers. It is also possible to replace more parts of the statement, or even to have the whole statement defined with a single variable, and to provide the complete SQL in a CHARACTER datalot.

Additional caution must be exercised when using dynamic SQL. Ensuring that a dynamic statement is syntactically and semantically correct is each user’s responsibility.

Switching to dynamic SQL does not impact Database Adapter operations including defining the Input and Output parameters and datalot mapping.

**Database Adapter configuration file**

An example of the generated configurator file is shown here:

```xml
<configform>
  <block title="variables" desc="Database Adapter in and out variables"
         hidden="false">
    <param name="var1" type="java.lang.String" value="" access="I"
           description=""/>
    <param name="var2" type="java.lang.Integer" value="" access="O"
           description=""/>
    <param name="var3" type="java.lang.Double" value="" access="IO"
           description=""/>
    <param name="var4" type="java.lang.Object" value=""
           access="HIDDEN" description=""/>
  </block>
  <block title="statements" desc="SQL Statement related info"
         hidden="true">
    <param name="sql.count" type="java.lang.Integer" value=""
           access="" description=""/>
    <param name="sql1.name" type="java.lang.String" value=""
           access="" description=""/>
    <param name="sql1.db" type="java.lang.String" value=""
           access="" description=""/>
  </block>
</configform>
```
Database properties file

When you create an application having one or more Database adapters in Progress Developer Studio for OpenEdge, it stores that database adapters’ connection properties in the OEBPS_WORKSPACE\com.savvion.studio\adapters\DBAdapters\db.properties file. When you publish such an application on Business Process Server, you must update the OEBPS_HOME\adapters\db\conf\db.properties file on Business Process Server as explained in the following section.

Updating the Database adapter properties file on the server

To update the Database adapter properties file on the server with the database properties of your Database adapter:

1. Open the OEBPS_HOME\adapters\db\conf\db.properties file in a text editor.
2. In the beginning of the file, increase the value of db.count by the number of Database adapters’ properties you want add from your application(s).
3. Open the OEBPS_WORKSPACE\com.savvion.studio\adapters\DBAdapters\db.properties file in a text editor.
4. Copy the group of database properties of your Database adapter that you want to add in the db.properties file on the server.
5. Paste the copied database properties at the end of the db.properties file on the server.
6. Increase the index for that group of database properties by one.
   For example, in the db0.name property, ‘0’ is the index for that property. If you increase the db.count to 2 in Step 2, then set the index to ‘1’ for the group of database properties you add.
7. Save the db.properties file on the server.

A sample database properties file is shown below:

```
db.count=1
db0.name=oebps
db0.type=openedge
```
### Working with DataDirect Cloud drivers in the Database Adapter

The Database Adapter extends support to the DataDirect Cloud drivers. You can make use of the Database Adapter to access and use data from the several Progress DataDirect data sources in your business application.

To use DataDirect Cloud drivers in the Database Adapter of your Business process:

1. Set up the data sources in Progress DataDirect.
   - For more information about setting up data sources, see the *Progress DataDirect* documentation.

2. Create a new or open an existing Business process file (SPT) with a Database Adapter that must be set up to use data from data sources in Progress DataDirect.

3. Configure the database adapter to specify DataDirect Cloud as your choice of database type and to enter the DataDirect Cloud connection details, and then to enter all the other configuration details in the **DBAdapter Configurator** dialog box.

4. Build the process to create artifacts associated with the Business Web flow after configuring the Database adapter and other work steps of your Business Web flow.

After successfully building your Business Web flow, you must update the database properties and server files before you can deploy the application. For more information on updating the properties and the server files, see the *Updating Database properties and server files* on page 53.

---

**Note:** For the latest Data Direct Cloud drivers, login to http://www.datadirectcloud.com and download the DataDirect Cloud drivers from the downloads page. After installing the latest DataDirect Cloud drivers, use the "ddcloud.jar" from "${INSTALL_LOCATION}/lib" to replace the "ddcloud.jar" in the following locations:

- `${DLC}\oebpm\server\lib\jdbc\datadirect\lib`
- `${DLC}\oebpm\modeler\eclipse\plugins\com.progress.openedge.pdt.bpm.jdbc.d2c_11.3.1\lib`

After replacing the "ddcloud.jar", restart the Process Modeler and the BP Server.

---

### Using the Database Adapter

When using the Database Adapter, special care must be taken to ensure that the adapter is configured properly, and that it operates correctly when your business process is published.

Upon installation, Progress Developer Studio for OpenEdge (or Business Process Modeler) is not automatically configured for use with a particular database. The Database Adapter needs a database connection in order to be configured properly.
When using the Database Adapter configurator for the first time, you will note that the default Business Process Server database - "oebps" - is not configured. One way to fix this is to define the database connection properties using the Database Adapter configurator as described in Defining Database connection on page 42. Another approach is to copy the db.properties file from the Business Process Server installation you are designing the process for. Copy the db.properties file from OEBPS_HOME\adapters\db\conf directory of the Business Process Server installation to OEBPS_HOME\workspace\.com.savvion.studio\adapters\DBAdapters.

If you are using the Database Adapter Configurator to add new database connections, you must ensure that the db.properties files contain the same data in Progress Developer Studio for OpenEdge (or Business Process Modeler) and the Business Process Server. You can copy the db.properties file from Progress Developer Studio for OpenEdge (or Business Process Modeler) to the Business Process Servers directory; just be careful not to overwrite some database connection that is present and used on the Business Process Server, but is missing in Progress Developer Studio for OpenEdge (or Business Process Modeler).

For more information on the content and format of the db.properties file, see the Database properties file section in Progress OpenEdge Business Process Modeler: User's Guide.

**Updating Database properties and server files**

When you create an application having one or more Database Adapters, it stores that database adapter's connection properties in the WORKSPACE\.com.savvion.studio\adapters\DBAdapters\db.properties file.

When you publish such an application on the server, you must update the OEBPM_HOME\server\adapters\db\conf\db.properties file on Business Process server and the OEBPM_HOME\jboss\server\ejbServer\deploy\bmjdbc\sbmcommondb-ds.xml file with the database properties and the data sources details.

To update database properties file on the server:

1. Copy the configurations of your Database Adapter from the db.properties file in the WORKSPACE\.com.savvion.studio\adapters\DBAdapters\location to the db.properties file on the server in the OEBPM_HOME\server\adapters\db\conf\location.

2. After copying the contents from the workspace location to the server location, ensure that you remove all the instances of “\” (backward slashes) from the db.properties file in the server location.

For instance, if you have added a Database adapter to your Business Web Flow and your db.properties file in your workspace has the following entry:

```properties
db1.name= DataDirect CloudNew
db1.type= DDCloud
db1.user=asmishra
db1.pass= encrypted password
db1.jndi= jdbc/DDCloudJNDI
db1.driver=com.ddtek.jdbc.ddcloud.DDCloudDriver
db1.url= jdbc:datadirect:ddcloud://das.ddcloud.progress.com;databaseName=[DATASET_NAME];
```

```java
{"db1\"\: {  "name": "DataDirect CloudNew",  "type": "DDCloud",  "user": "asmishra",  "pass": "encrypted password",  "jndi": "jdbc/DDCloudJNDI",  "driver": "com.ddtek.jdbc.ddcloud.DDCloudDriver",  "url": "jdbc:datadirect:ddcloud://das.ddcloud.progress.com;databaseName=[DATASET_NAME]"}}
```
Update the db.properties file in the server, which might have other database properties in it, as follows:

```
db.count=2
db0.name=MYORACLEDB
db0.type=oracle
db0.user=ora_user
db0.pass=r9lXm5ut6A=
db0.jndi=jdbc/MyOracleDB
db0.driver=oracle.jdbc.driver.OracleDriver
db0.url=jdbc:oracle:thin:@server1:1521:orcl
```

```
>db0.additional=
db1.name= DataDirect CloudNew
db1.type= DDCloud
db1.user=asmishra
db1.pass= encrypted password
db1.jndi= jdbc/DDCloudJNDI
db1.driver=com.ddtek.jdbc.ddcloud.DDCloudDriver
db1.url= jdbc:datadirect:ddcloud://das.ddcloud.progress.com;dbname=[DATASET_NAME];
```

```
db1.additional=
```

**Note:** The server file may have existing database adapters, do not overwrite any existing properties, append the new one and give it the next higher number (db1 in the above example).

3. Increase the `db.count` by one whenever you add a new database property detail in the db.properties file, and then append to the end of the file, the new database's properties.

4. Update the `OEBPM_HOME\jboss\server\ejbServer\deploy\bmjdbc\sbmcommondb-ds.xml` file with the newly added DDCloud driver Data Source information.

The following shows a sample entry of a DataDirect Cloud driver Data Source in the `sbmcommondb-ds.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<datasources>
  <local-tx-datasource>
    <jndi-name>jdbc/DDCloudJNDI</jndi-name>
    <connection-url>jdbc:datadirect:ddcloud://das-qa-int.palette-test.progress.com;dbname=MyDataSet</connection-url>
    <driver-class>com.ddtek.jdbc.ddcloud.DDCloudDriver</driver-class>
    <use-java-context>false</use-java-context>
    <user-name>asmishra</user-name>
    <password>password</password>
    <min-pool-size>5</min-pool-size>
    <max-pool-size>40</max-pool-size>
    <idle-timeout-minutes>5</idle-timeout-minutes>
    <blocking-timeout-millis>5000</blocking-timeout-millis>
    <prepared-statement-cache-size>50</prepared-statement-cache-size>
    <transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>
  </local-tx-datasource>
</datasources>
```

5. Restart the servers from First Steps Console (Start > All Programs > Progress > OpenEdge 11.X > BP Server > First Steps Console) for the changes to take effect.
Using the DBAdapterDemo sample application

The DBAdapterDemo sample application demonstrates the use of the multiple row selection option in the Database Managed Adapter, presenting the results in a JSP using BPM Webflow’s displayResultSet tag.

The demo application uses the Database Adapter to retrieve data about the installed applications, active process instances and tasks.

You can install the application from Business Process Portal if you have the administrative privileges. Go to Administration > Applications > BPM Webflow and select DBAdapterDemo. For running this application, you should have some active BP Server processes so that valid data is displayed.

Tutorial 1: Using the Database Adapter

The following tutorial provides step-by-step instructions for setting up a sample database table, configuring the Database Adapter, and then designing, deploying, and using it in a business application.

Before starting this tutorial, read Understanding the Managed Adapter framework on page 19 and Working with the Database Adapter on page 41.

Before you begin designing the sample application, you will need to know the following information about the database your Business Process Server installation uses.

• Database type.
• Username and password.
• Driver class.
• Database URL.

To access this information, go to db.properties file under the OEBPS_HOME\adapters\db\conf directory.

You need to copy the OEBPS_HOME\adapters\db\conf\db.properties file (if existing) from the Business Process Server installation machine to the conf directory. If the Database Adapter has been already used in Business Process Server, this file will contain a list of the defined databases for the Business Process Server installation.

Note: A frequent slip in the Database Adapter configuration is to forget updating the list of output parameters when you update your SQL statements. For example, if you initially had a statement returning two variables V1 and V2 and later changed the statement to return only V1, you may notice that the V2 output still appearing in the map configurator window. In this case, you most probably forgot to remove the V2 output variable from the DBAdapter's variable list. Click the Variables button in the Database Adapter configurator window and remove all the variables that are not used.

Step 1: Creating the customer database table

The following table shows a sample customer database.
Table 12: Sample customer database

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>PHONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>George Smith</td>
<td>(111) 555-1111</td>
</tr>
<tr>
<td>101</td>
<td>Mike Green</td>
<td>(222) 555-2222</td>
</tr>
</tbody>
</table>

You can use your existing table or create another one as per your requirement. For a practical application, you will also need to configure your own datasource. Refer to the application server documentation on how to set up a datasource.

Step 2: Creating the process flow

Create a BP Server application. Define the type of process flow that is shown in the following figure.

Figure 4: Creating the Process Flow

Use the workstep icons in the Task pane to define the process flow. These worksteps will be used to perform the following functions:

- **Start** Workstep. Starts the process flow and supplies a customer ID.
- **Query** Workstep. Calls the Database Adapter, which will look up the personal information from the customer database.
- **ShowInfo** Workstep. This displays the customer record from the database.
- **End** Workstep. This is the end of the process flow.

Step 3: Defining the dataslots

Click the **Dataslots** tab and define the dataslots to store the customer information - CustID, CustName, CustAddr and CustPhone. The CustID is of INTEGER numeric type, the CustName, CustAddr, and CustPhone dataslots are CHARACTER datatype.

Step 4: Defining start workstep properties

You must next define the properties of each workstep.

1. Click the **Start** workstep to open its **Properties** view.
2. Click the **Fields** tab. The list of dataslots for the Start workstep initially is empty.
3. For this workstep, you need to add the CustID dataslot. Click **Add**, opening the **Select Dataslots** dialog window and select CustID from the list and then click **OK**. The CustomerID dataslot appears in the dataslot list, as shown in the following figure.
4. Click the **General** tab and confirm that the workstep is using the automatically generated Default presentation form, as shown in the following figure.
Step 5: Defining ShowInfo workstep properties

To define ShowInfo workstep properties:

1. Double-click the ShowInfo workstep to open its Properties view.
2. Click the Fields tab. Configure the workstep properties, this time selecting all the dataslots — CustID, CustName, CustAddr and CustPhone.
3. Click the General tab and confirm that the workstep is using the automatically generated Default presentation form.
4. Click OK, and the ShowInfo workstep is completely defined.

Step 6: Defining performer for query workstep

To define a performer for query workstep:

1. Click the Assign Participants link in the Tasks pane, and expand the Adapters > Managed > DBAdapters folder.
2. Drag the GenericDBAdapter icon into the blank workspace, and a workstep appears with a DBAdapter icon. The next step is to configure the Database Adapter.

Step 7: Configuring the Database Adapter

To configure the database adapter:

1. From the Properties view of the Query workstep, click the Configuration tab and then click Configure. The Database Adapter Configurator dialog box appears.

2. Click Add to add a new statement. The SQL Statement dialog box appears.

   Note that the Database Parameters details are blank, as the database connection is not yet configured.

   a) Click Choose a Database. The Database Definitions dialog box appears.

      You can define a new database connection.

   b) Click the Add button. The Database Parameters dialog box appears.

   c) Enter the appropriate values for your configuration. Note that, for this demonstration, jdbc.SBMCommonDB is used as the datasource. However, in practice, you need to define your datasource with your own connection pool.

      The database name must be the same as the value of the db0.name parameter in the OEBPS_HOME\adapters\db\conf\db.properties file. The default value of this parameter is oebps. If you want to change it, then update the db.properties file appropriately before specifying it in the Database Parameters dialog box.
d) Click **OK**. The **Database Definitions** dialog box shows the SampleDatabase.
e) Select SampleDatabase, and click **OK**. The **SQL Statement** dialog box now displays the parameters for the selected database.

3. Click **Connect**. After a short delay, the Status pane shows the message: "Successfully connected to database: SampleDatabase".

   In case of errors, refer to the error messages in the Status pane. We recommend that if an error occurs, check that:

   - You entered all the connection parameters correctly.
   - The database server is running and accessible.
   - The database drivers are in the correct place.

   **Note:** You need to define the database connection only once. For future use, you only need to select the existing connection from the Database Definitions list.

---

**Step 8: Defining the SQL statement**

To define SQL statement:

1. In the **SQL Statement** dialog box, enter the statement name. For example, FindCustomerInfo.
2. Click the **SQL Statement** tab.
   a) Choose the SQL statement type (for information on types, see Defining SQL statements on page 43). Since you plan to query the database, and are not doing any modification, choose **Select**. A button named **Choose Tables** is enabled in the **Clauses** panel. Click this button to open the **Choose Table(s)** dialog box.
   b) Select the table CUSTOMER_INFO from the list (we created this table in Step 1: Creating the customer database table on page 55), and click **OK**. The SQL Statement tab in the **SQL Statement** dialog box now displays the CUSTOMER_INFO details.

   You have already selected the table, and now need to select the columns and the lookup condition.

c) Click **Choose Columns**. A list of all available columns in the table appears in the **Choose Columns** dialog box.

d) Select the columns you need to get the information from — in this example, CUSTOMER_INFO.NAME and CUSTOMER_INFOPHONE.

e) Click **OK**. The selected columns now displayed in the **SQL Statement** dialog box.

3. In the SQL Statement dialog box, click **Add Conditions** to display the **Compose a Condition Clause** dialog box.
   a) Since you will be selecting records from the table based on customer ID, choose CUSTOMER_INFO.ID from the Database Columns drop-down list.
   b) Click **Append a Column**, to add the selected column to the Where clause.
   c) From the **SQL Operators** drop-down list, select "=" and click **Append an Operator**, adding the selected operator to the Where clause. After you have completed the composition, the condition clause appears.

   **Note:** You can select the **Enable Edit** check box to edit the condition directly. Once you are familiar with the Database Adapter operation, you may prefer to use the expert mode.
Note that the condition is incomplete. It reads:

\[
\text{CUSTOMER\_INFO\_ID} = \]

You need to define the right side of the equation to complete it. Since the value will not be constant, you can put a variable at this place.

4. Click **Append a Variable**. A list of the existing variables (if any) are displayed in the **Variable Definitions** dialog box.
   a) Click **Add** to create a new variable. The **Variable Editor** dialog box appears
   b) Enter **ID** for the variable name, as it will store the customer ID number.
   c) Select **Integer** from the Type list for the data type.
   d) From the Access list, select **Input**, as it will be passed as input to the Database Adapter. Enter 0 (zero) for the default value.
   e) Click **OK**. The ID variable now appears in the variable list in the **Variable Definitions** dialog box.

5. Select the "ID" variable and click **OK**. The condition is now complete:

\[
\text{CUSTOMER\_INFO\_ID} = \text{@ID}
\]

6. Click **OK** to close the condition editor. The **SQL statement** dialog box displays the complete statement.

**Step 9: Defining the output parameters**

You have already defined the input parameter, the variable **ID**, used to supply the Database Adapter with the customer ID number. Now you can define the list of output parameters - all the data that the Database Adapter should return to your business process as the result of the query.

Before defining the output parameters, the Database Adapter Configurator must know the format of the result set from the SELECT statements, or in other words - the query result meta-data.

**To define the output parameters:**

1. In the **SQL Statement** dialog box as shown in **Step 8: Defining the SQL statement** on page 58 above, click **Get Result MetaData** to display the **Query Tester** dialog box. The list of our input variables currently contains just one variable, **ID**, that you have defined before.

   Note that the Query Result pane is blank. Before executing the statement in the SQL Statement Preview pane, you need to define values of the input parameters; in this case, **ID**.

2. In the **Value** box, enter the customer ID of one of the existing users, using one of the IDs you had created in **Step 1: Creating the customer database table** on page 55.

3. Click **Execute**. The **Query Tester** dialog box shows the results of the query in the Query Result pane.

   The Query Result pane shows the personal data for the selected customer.

   **Note:** If the Query Result pane is empty, then the ID may not be specified correctly. You can connect to the database and check for validity of the ID.

4. Click **OK** to close the **Query Tester** dialog box. You will see that the **Output Parameters** tab in the **SQL Statement** dialog box is now enabled.
5. Click the **Output Parameters** tab. The list of the columns selected from the table is displayed.

   Note that the **Map To** column is empty, indicating that none of the columns is mapped to an output parameter. Mapping to an output parameter is the way to get data out of the Database Adapter. The adapter returns a list of output variables to your business process and now you must define the table fields in which the output variables are stored.

   a) Click the **ellipses** ( ) button in the "NAME" row. The **Variable Definitions** dialog box appears.

   b) Click **Add** to define a new variable; in this example, **name**. The **Variable Editor** dialog box now appears.

      Note the default value should be set to **unknown**, since this value will be displayed when the query returns no customer. For example, if the customer ID is wrong, the Customer Name field will display "unknown".

   c) Click **OK**. The new variable name will appear in the variable list of the **Variable Definitions** window.

   d) Select **name** and click **OK**. The **Output Parameters** tab shows the Map To for the NAME output parameter set to "@name", indicating that the column is mapped to the variable **name**.

6. Repeat the above step for the PHONE row and map it to **phone**.

   The final **SQL Statement** dialog box appears.

   Note that each of the columns has been mapped to the corresponding output variable. The Database Adapter configuration is now complete.

7. Click **OK** to close the **SQL Statement** dialog box.

8. Click **OK** in the **DBAdapter Configurator** dialog box to save the updated configuration. This action also opens the Map Configurator, which is described in the following section.

**Step 10: Mapping Adapter inputs / outputs to dataslots**

The Database Adapter configuration is complete with input and output parameters. The next step is to define the association between the adapter input/output parameters and the dataslots for the application defined in **Step 2: Creating the process flow** on page 56.

**To map adapter inputs / outputs to dataslots:**

1. Open the **Dataslot Mapping** dialog box by clicking **Change mapping** from the **Configuration** tab of the **Properties** view of the Query workstep.

2. You can map the input parameter **ID** to the **CustID** dataslot, the **name** parameter to the **CustName** dataslot, the **address** parameter to the **CustAddr** dataslot, and the **phone** parameter to the **CustPhone** dataslot.

3. Click **OK** to return to the **Properties** view of the Query workstep.

4. Click **OK**. Your business process is now completely defined.

5. Save the business process under the name **CustomerInfo**.

When the process is started, the user is presented with a prompt to enter a customer ID (Start workstep). The Database Adapter (Query workstep) uses this ID to obtain the customer name and phone number from the database. This data is then displayed (ShowInfo workstep).
Step 11: Publishing the completed application

If you are using Progress Developer Studio for OpenEdge, publish the application from the Project > Publish. You also need to copy the db.properties file, updated by the AdapterConfigurator to the Business Process Server's OEBPS_HOME\adapters\db\conf directory.

Note: Ensure that you do not overwrite any existing database definitions. For more information on Updating Database properties and server files, see Updating Database properties and server files on page 53.

Step 12: Running the application

To run the application:

1. After the application is published, login to Business Process Portal. From the Home tab, select Applications > CustomerInfo. The start page for the CustomerInfo application appears.

2. Enter the customer ID, and click Create. A new instance of the application is created, the Database Adapter is executed, and the task with the requested customer information will be available in your task list.

3. Click the My Tasks tab, and tasks are listed in the Tasks page.

4. Click the ShowInfo name in the Task column. The requested customer information is displayed.
Email Managed Adapter

The Email Adapter enables you to generate and send e-mail messages from your business process. With the Email Adapter, you can create messages that can contain multiple MIME parts with text, HTML, XML, and other types of content, including attachments from files.

For details, see the following topics:

• Email Adapter features
• Preconfiguring the Email Adapter
• Working with the Email Adapter
• Sending messages to multiple recipients
• Using the Attachment plug-in
• Troubleshooting the Email Adapter

Email Adapter features

The features of the Email Adapter include:

• Sending multipart MIME e-mail messages.

• Generating messages that contain parts in all formats supported by the standard format plug-ins. For information on format plug-ins, see Using the Text Format plug-in on page 80 and other plug-in descriptions in File Managed Adapter on page 77

• Generating messages that contain attachments from files.
• Sending messages to multiple recipients in a single workstep, while customizing the message content for each recipient.

• Using the plug-in development kit to customize formats, as described in Writing custom format plug-ins topic in *Progress OpenEdge Business Process Modeler: User’s Guide*.

• Providing a custom Adapter Configurator GUI.

Preconfiguring the Email Adapter

Before being used, the Email Adapter requires some information about your mail server. Although you can supply this information every time when you use the Email Adapter in one of your business processes, it may be more convenient to preconfigure the adapter. By preconfiguring the Email Adapter, you will not have to enter the same information every time.

**To preconfigure the Email Adapter:**

1. From Tools menu, select **Managed Adapters** to open the Managed Adapter Browser (available only from Progress Developer Studio for OpenEdge) dialog box. In this browser, expand **EmailAdapters** and double-click the **GenericEmailAdapter** to display the **Email Adapter Configurator** dialog box.

2. Click the **Content** tab.

3. On the **Content** tab, click the **Subject** tab and add a template or import a template by clicking the **Add** and the **Import**. Click **Edit** to edit the template.

   **Caution:** The maximum number of characters allowed for a subject line is 998 and anything above this limit will not appear in the subject line. Do not use multi-line templates for subject. Any line breaks that you enter will be replaced with spaces and the entire subject will appear in one line.

4. Click the **Body** tab and add a template or import a template by clicking the **Add** and the **Import**. Click **Edit** to edit the template.

   **Note:** In addition to configuring, you can use the Managed Adapter Browser to perform functions like creating a copying, renaming, deleting, as well as importing and exporting the configuration information. For more information, refer to the "Using the Managed Adapter Browser" section of the *OpenEdge Getting Started: Developing BPM Applications with Developer Studio*.

5. Click the **Configuration** tab and enter the mail server information as detailed in the following steps.

6. Set the Mail Host parameter to the name of your SMTP server. If your environment requires some special settings, you may enter them in the **Extra Properties** text area.

7. If your email (SMTP) server requires authentication, enter a user name and password in the appropriate fields.

   **Note:** In most cases, you may only need to modify the Mail Host parameter to the name of your SMTP server. For more information, see the full list of SMTP server properties found at Sun’s javax.mail documentation at the following URL address: [http://java.sun.com/products/javamail/javadocs/index.html](http://java.sun.com/products/javamail/javadocs/index.html).

8. In the **Language** field, enter the language that you wish to specify for this template.

   If you do not specify a language here, you must select the language in Dataslot Mapping.
9. Typically, in case of a serious error (for example, inaccessible mail server or an invalid e-mail address format) the E-mail Managed Adapter throws an exception, and suspends the adapter workstep.

Select the Errors as Output check box if you do not want to suspend the workstep. It also presents a parameter with the error description in the adapter outputs. You can map this output to a string dataslot. For more information about mapping dataslots, refer to Defining mapping on page 69. In the process, you can evaluate this dataslot value, and if it is not empty, decision about error handling can be taken as per your requirements.

10. In the Extra Properties text box, you can set the following parameters.

a) You can use the allow.null.cc and allow.null.bcc parameters to specify what the adapter should do if cc or bcc has a null value. If the "Cc" or "Bcc" field is mapped to a null CHARACTER dataslot, the Email Adapter ignores it. However, you can change this default behavior. To do so, set the following parameters to false.

```plaintext
allow.null.cc=true
allow.null.bcc=true
```

The default value of this parameter is true.

b) You can use the mail.dispatch parameter to specify whether the adapter should send a single message to all the recipients or multiple messages individually to each recipient. The syntax of this parameter is as given below:

```plaintext
mail.dispatch=[single|multiple]
```

If you set this parameter to single, the adapter sends a single message to all recipients listed in the "To:" field. However, this disables the message customization as described in Sending messages to multiple recipients on page 70. The default value of this parameter is multiple.

c) You can use the mail.ignore parameter to suppress errors generated by a specified plugin. The syntax of this parameter is as given below:

```plaintext
mail.ignore.<plugin_name>.errors=[true|false]
```

When a plugin encounters a problem during invocation, the adapter is suspended by default. If you set this parameter to "true" for a particular plugin, the adapter will not be suspended, and the message will be sent without the message content causing the error.

d) You can use the mail.ignore.attachment.errors parameter to send the message without attaching a document. The syntax of this parameter is as given below:

```plaintext
mail.ignore.attachment.errors=[true|false]
```

11. Once you have configured your mail server, click OK to close the Adapter Configurator.

From now on, the configuration you set will be preset by default every time you use the Email Adapter in a business process.

---

**Working with the Email Adapter**

The Email Adapter is a Managed Adapter and its usage is standardized along the same lines as the other Managed Adapters. For more information, see the adapter configuration and mapping framework described in Understanding the Managed Adapter framework on page 19.
Configuring the Email Adapter

You can assign the Email Adapter to an Adapter workstep in your process by performing the following procedures:

1. Click the **Assign Participants** link in the Tasks pane and then open the **Adapters > Managed > EmailAdapters** folder.

2. Drag the **GenericEmailAdapter** icon to the Content pane.
   
   Alternatively, you can right-click the Adapter workstep and select **Change Performer** from the pop-up menu. Select **GenericEmailAdapter** from those listed in the **Change Performer** dialog box and click **OK**.

3. From the **Properties** view for the workstep, click the **Configuration** tab. This tab includes buttons for defining the adapter configuration, mapping, and specifying advanced properties.

4. Click **Configure** to open the **Email Adapter Configurator** dialog box.

5. Enter information in the fields displayed in the **Message** tab, as indicated in the following table:

<table>
<thead>
<tr>
<th>Field</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>From</td>
<td>Type the Sender e-mail address you want your messages to originate from. Make sure you enter a valid e-mail address, so you are able to receive any user replies or system messages (for instance, Delivery failure notifications). If you leave this field empty, you are prompted later to map it to a dataslot.</td>
</tr>
<tr>
<td>To</td>
<td>You can type one or more Recipient e-mail addresses, to whom you want to deliver the email message. If you enter more than one address, use commas as separators, example: &quot;<a href="mailto:user1@company.com">user1@company.com</a>, <a href="mailto:user2@company.com">user2@company.com</a>, ...&quot;. If you enter more than one address here, a number of separate messages are sent—one to each of the recipients. If you leave the 'To:' field empty, you are prompted later to map it to a dataslot. You can map this value to either a CHARACTER dataslot, containing one or more e-mail addresses in the format specified above, or to a LIST dataslot.</td>
</tr>
<tr>
<td>Cc / Bcc</td>
<td>You can type a list of e-mail addresses to deliver a carbon copy (Cc) or blind carbon copy (Bcc) of the message. Note that if you are sending a message to a number of To-recipients, the Cc- and Bcc-recipients receives one message per every one To-recipient. The reason for this is that the message content may be customized on a pre-recipient basis and may vary for every To-recipient.</td>
</tr>
<tr>
<td>Field</td>
<td>Information</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>Type the message subject here. If you leave this field empty, you can enter the subject in the content tab. If you don’t enter the subject in the content tab either, you are prompted to map it to a dataslot. You can also use variables to customize your subject. Each variable can be mapped to a dataslot when the Adapter dataslot mapping is defined. For example, if you set the Subject to: Hello $customerName, an adapter input named &quot;customerName&quot; is created. This input is then be available for mapping from a dataslot. <strong>Note:</strong> However, if you leave the Subject empty here and if you do not specify the subject in subject tab on the content tab, and later map it to a dataslot, using variables in the mapped value is not allowed. In other words, if you map the subject to a CHARACTER dataslot, containing the value &quot;Hello $customerName&quot;, the variable &quot;customerName&quot; are not recognized. If there is a matching subject template in content tab with the specified language, then the template is picked. If not, the subject specified in the message tab will be picked up.</td>
</tr>
<tr>
<td>Body</td>
<td>The table in the Body: field details the content of the message. Use the buttons on the right to define a list of MIME parts (see Defining the message body and configuring the format plug-ins topic in the Progress OpenEdge Business Process Modeler: User's Guide). The content of each part are generated by a format plug-in, configured separately in the &quot;Mime Body&quot; tab. Plug-in configuration is discussed in detail in the Preconfiguring the email adapter topic in Progress OpenEdge Business Process Modeler: User's Guide. To initially configure, just select the appropriate plugin, corresponding to the data format you want the MIME part to contain, and specify the appropriate mime-type in the &quot;Data Type&quot; column.</td>
</tr>
</tbody>
</table>

6. The information you enter in the **Content** tab varies, depending on the format plug-ins you selected in the **Body text area** in the **Content** tab.

7. The **Configuration** tab includes parameters specifying the mail server information and the **Language** field. If you have already preconfigured the adapter, you do not have to modify the connection data in this tab. Refer to Preconfiguring the email adapter topic in Progress OpenEdge Business Process Modeler: User's Guide for information on how to use the **Configuration** tab and how to specify server defaults. If you need to have special configurations for the given business process you are designing, you can modify the parameters in the **Configuration** tab. The changes, however, are limited to your current business process.

### Defining messages

The Email Adapter can generate MIME multipart messages. Each part can contain different types of data and be generated by a different format plug-in.

**To define a multipart message:**

1. Click Add in the **Message** tab of the **Email Adapter Configurator**. Body is automatically added in the form of a row to the table.

   **Note:** If you choose the Format plugin as Attachment, the email will have only the Attachment and no other content.

2. Click the new row in the **Format Plugin** column, and choose one of the displayed options. These options are:
### Table of Plug-ins and Descriptions

<table>
<thead>
<tr>
<th>Plug-in type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Velocity Template</td>
<td>For plain text, XML, HTML, PostScript and other formats, all generated with the Velocity template engine. For more information on this plug-in, see Using the Velocity Template plug-in on page 81.</td>
</tr>
<tr>
<td>Text</td>
<td>Generates simple text messages. The message content is usually mapped from a CHARACTER dataslot. For more information on this plug-in, see Using the Text Format plug-in on page 80.</td>
</tr>
<tr>
<td>XML</td>
<td>Populates predefined XML forms using XPath expressions. For more information on this plug-in, see Using the XML Format plug-in on page 83Using the XML Format plug-in on page 11.</td>
</tr>
<tr>
<td>Attachment</td>
<td>Adds a file attachment to the message. The file can either be from the server’s file system or can be mapped from a Document dataslot. For more information on this plug-in, see Using the Attachment plug-in on page 73. If you choose the Format plugin as Attachment, the email will have only the Attachment and no other content.</td>
</tr>
</tbody>
</table>

**Note:** You can also use any custom format plug-ins you may have already developed. For more information on developing plug-ins for custom data formats, see Writing custom format plug-ins topic in Progress OpenEdge Business Process Modeler: User’s Guide.

3. After you have selected a Format Plug-in option, a default MIME type is automatically added to the MIME Type column. Click in this column to modify the data type.

4. After you add Body and Attachment, use the **Move Up** and **Move Down** buttons to place the parts in the proper sequence. Use the **Remove** button to remove an entry from the table.

### Defining the message body and configuring the format plug-ins

After defining the structure of the message, you must define the message contents by configuring the format plug-ins used in the message. You can add velocity templates or XML data etc.

**To define message contents by configuring format plug-ins:**

1. Open the **Content** tab, which contains two tabs, the **Subject** and **Body** tabs, one for each format plug-in defined when constructing the body of the message. For **Attachment**, there is no tab as it does not require any configuration at this stage.

2. The **Edit Template** window allows you to select the Language.
   a) The template contains variables that are replaced with real-time data. Click **Extract Variables** to display the variables in the text area below.
   b) Click **Clear** to clear all data from the dialog box.
   c) Click **Load Template** to add a template to the message.
You can click Add to add a new template. Click Import to import templates. Click Edit to edit the template.

3. Click OK to complete the configuration of the Email Adapter. This action also automatically opens the Dataslot Mapping dialog box.

**Using AdapterConfigurator to configure Email Adapter instances**

Once the Email Adapter is installed, you can configure a particular adapter instance using the AdapterConfigurator, a GUI-based component and specific to an adapter or a set of adapters. The configurator allows you to create an adapter configuration file in a standardized format common for all the Managed Adapters. The AdapterConfigurator can also be defined separately for each adapter instance, and each adapter can also have its own custom configurator.

You can start the AdapterConfigurator using the following command:

```
AdapterConfigurator adapterCategory adapterName
```

For the adapter configurator, you need to specify the adapter’s category and instance names. If the adapter has an adapter configurator defined in this framework, the configurator GUI will be opened. If the adapter has not been defined in this framework, the GUI will not appear. When the configuration is completed, the configurator saves the adapter configuration in the `config.xml` file in the `maps` directory within the adapter instance directory.

**Defining mapping**

After configuring the Email Adapter, you must define the mapping of your process dataslots to adapter inputs/outputs. When you click OK in the Email Adapter Configurator, the Dataslot Mapping dialog box appears. The number of tabs displayed in this dialog depends on the number of adapter components (or parts) that were defined during configuration. In this example, we defined three parts during in the Email Adapter Configurator, and there are three tabs—Subject, Body and Attachment—displayed in addition to the Message tab.

**To define mapping:**

1. In the Message tab, map the dataslot to the To parameter to provide the recipient’s e-mail address. You can also map dataslots to the Cc and Bcc fields — either to a CHARACTER dataslot or LIST dataslot, or to a constant that is defined by directly typing a comma-separated list of addresses in each of the fields.
2. In the Subject tab, map the variables defined during the configuration of Velocity Template to relevant dataslots.
3. In the Body tab, map dataslots to previously defined parameters — the values from the dataslots are used to generate the body of the message; that is, the order form.
4. In the Attachment tab, map the Document dataslot to the Attachment parameter. For more information, see Using the Attachment plug-in on page 73.
5. With dataslot mapping completed, click OK to complete the Email Adapter definition. The Email Adapter is now ready to use.

If the "Cc" or "Bcc" field is mapped to a null CHARACTER dataslot, the Email Adapter ignores it. However, you can change this default behaviour. To do so, on the Configuration tab of the Properties view, click Configure to open the Email Adapter Configurator window. In this window, click the Configuration tab. On this tab in the Extra Properties text box, by default, the following options are set to ‘true’, set these options to ‘false’.

```
allow.null.cc=true
allow.null.bcc=true
```
Note: If any or both of these options are set to 'false' the adapter is suspended if a non-initialized CHARACTER dataslot is mapped to the Cc: and/or Bcc: fields.

Sending messages to multiple recipients

The Email Adapter is capable of sending multiple messages to a group of recipients, all in a single workstep. Each message may be customized for the specific recipient. The following rules govern using the Email Adapter to send messages to multiple recipients:

• For each recipient in the “To:” list, a separate email message is sent. You can send a number of messages without the recipients seeing each other’s addresses. If you want only one message to be sent, use the Cc: or Bcc: field. You must have at least one address in the To: field, however.

If the mail.dispatch parameter is set to "single" as described in Preconfiguring the email adapter topic in Progress OpenEdge Business Process Modeler: User's Guide, then a single message is sent to all the recipients listed in the "To:" field, as the message customization is disabled.

• If the "To:" field is mapped to a LIST dataslot (but not a CHARACTER dataslot, containing a list of addresses), the message body is customized for each recipient in the list. This means that if any of the other adapter inputs are mapped to a LIST dataslot, the corresponding values from those dataslots are used for each recipient’s message. For example, if you have a LIST dataslot named "TO" with the following values:

   joe@company.com, albert@company.com, jeff@othercompany.com

and another LIST dataslot, "NAMES", that contain the names of the following people:

   Joe Black, Albert Green, Jeff White

when the "To" field is mapped to the "TO" dataslot, and a variable (say $customerName) in the message body text is mapped to the "NAMES" dataslot, the message sent to joe@company.com has the name “Joe Black” as value of $customerName, the one sent to albert@company.com has the name as “Albert Green”, and so on.

At the same time, if you have another variable used in the message body (for example, $announcementDate), and this variable is mapped to a non-LIST dataslot (for example a Date dataslot), the value is the same for all messages sent to the different users.

The same rules applies to any variables in the message subject.

Note: If you are sending the message to a number of "To"- recipients, and at the same time have some addresses in the "Cc:" or "Bcc:" fields, each of the "Cc"- or "Bcc"-recipients receives a number of messages—one message per "To"-recipient.

Using a LIST dataslot in the Email Adapter

The following use case example demonstrates how lists and scalars are treated in List mode; that is, when the "To:" field of an Email Adapter is mapped to a LIST dataslot.

In this example, the goal is to send notification messages to the winners of a lottery. Assume we have the data about the winners, the prizes, and the lottery phone number, in the following four dataslots:
Table 13: Dataslots used in the use case example

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Dataslot value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDRESSES</td>
<td>LIST of strings</td>
<td>[&quot;<a href="mailto:joe@company.com">joe@company.com</a>&quot;, &quot;<a href="mailto:albert@company.com">albert@company.com</a>&quot;, &quot;<a href="mailto:jeff@othercompany.com">jeff@othercompany.com</a>&quot;]</td>
</tr>
<tr>
<td>NAMES</td>
<td>LIST of strings</td>
<td>[&quot;Joe&quot;, &quot;Albert&quot;, &quot;Jeff&quot;]</td>
</tr>
<tr>
<td>PRIZES</td>
<td>LIST of lists</td>
<td>[&quot;36in TV&quot;, &quot;satellite radio&quot;],</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[&quot;binoculars&quot;, &quot;handheld GPS&quot;],</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[&quot;a penny&quot;, &quot;a dime&quot;, &quot;and a quarter!&quot;]</td>
</tr>
<tr>
<td>PHONE</td>
<td>CHARACTER</td>
<td>&quot;(408) 555-1234&quot;</td>
</tr>
</tbody>
</table>

Configuring the Email Adapter in List mode

To configure the Email adapter in List mode using the Velocity template:

1. In the Message Body tab of the Email Adapter Configurator dialog box, write the message.

   For Example:

   ```velocity
   Hello $ (Name),
   You just won the following:
   #foreach ($(p)in $ (PRIZES))
   - $(p)
   #end
   Please, call $(PHONE) to claim your prize.
   ```

   In the above example, ${NAME} will be substituted with the winner's name.

   The foreach loop will be expanded to a list of the prizes won. For this to happen, the ${PRIZES} variable must contain the List of prizes won by the given person.

   The ${PHONE} variable is replaced with the phone number to call, which is the same for all messages sent.
Finally, $P$ is an internal variable used in the loop that should not be mapped to a dataslot.

2. Click the **Extract Variables** button to list all four variables, NAME, P, PRIZES, and PHONE. As mentioned, P is an internal variable, so you can remove it from the list.

3. Complete the rest of the adapter configuration, leaving the "To:" field in the **Message** tab empty (so it will be mappable to dataslot) and close the **Email Adapter Configurator** dialog window.

### Mapping the Email Adapter in List mode

The following section describes how to define the dataslot mapping for an Email Adapter when the "To:" field of the Email Adapter is mapped to a LIST dataslot.

**To map the Email adapter in List mode:**

1. In the **Message** tab of the **Dataslot Mapping** dialog box, select the ADDRESS LIST dataslot to supply values for the "To:" field.

   Mapping the ADDRESS LIST dataslot to the "To:" field ensures that when the Email adapter is invoked, it will switch to List mode, extracting single elements from all LIST dataslots sent to the message body.

2. In the **Part 1: Velocity Template** tab, define the dataslot mapping as below:

   The $\{NAME\}$ variable will be mapped to the NAMES LIST dataslot. Because the adapter is in List mode, each time a message is sent, the corresponding element of the NAMES dataslot will be used.

   The same procedure applies to the $\{PRIZES\}$ variable and associated parameter. Note, however, that while the NAMES dataslot was a LIST of Strings, the PRIZES dataslot is a LIST of Lists, so each time a message is sent, the $\{PRIZES\}$ variable will contain the List of prizes won by the given person.

   Finally, the $\{PHONE\}$ variable is mapped to the PHONE dataslot. This is a simple CHARACTER dataslot and will be the same for each message.

   If everything is as expected, when the adapter is invoked, Joe will receive the following message (refer to **Table 13: Dataslots used in the use case example** on page 71, for information on dataslot names, types, and values):

   Hello Joe,
   You just won the following:
   - 36in TV
   - satellite radio
   Please, call (408) 555-1234 to claim your prize.

   Jeff, on the other hand, will see the following email:

   Hello Jeff,
   You just won the following:
   - a penny
   - a dime
   - and a quarter!
   Please, call (408) 555-1234 to claim your prize.

   Note that all this is possible only when the Email Adapter is invoked in List mode, i.e., when a LIST dataslot is mapped to the "To:" field. If we had mapped this field to a CHARACTER dataslot containing a comma-separated list of e-mail addresses, the adapter would have still send a separate message to each recipient, but the message content would be exactly the same for all users. (In particular, this template would not have worked properly, because the entire lists, and not just the list elements, would be passed to the NAME and PRIZES variables).
Using the Attachment plug-in

You can use the Attachment plug-in available in the Email Managed Adapter to attach a file to the e-mail message. The Attachment plugin allows the following mappings:

Table 14: Mappings Supported by the Attachment plug-in

<table>
<thead>
<tr>
<th>Mapping type</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document dataslot</td>
<td>The selected dataslots are used to supply the attached document.</td>
</tr>
<tr>
<td>CHARACTER dataslot</td>
<td>The CHARACTER dataslot should contain the location of the file on the local (appserver) file system. The file is loaded and attached to the email message.</td>
</tr>
<tr>
<td>Constant</td>
<td>If every message has the same attachment, and the attachment is present on the local (appserver) file system, you can directly type in the map configurator dialog the fully-qualified name of the file</td>
</tr>
</tbody>
</table>

If you want to attach more than one attachment, you can use the "Attachment" plugin several times in the same message.

**Note:** You cannot use the Email adapter to send multiple file attachments stored in a bundle Document dataslot.

To send an attachment with the e-mail message:

1. Double-click an Email Adapter workstep to open its Properties view. Open the Configuration tab and click Configure to open the Email Adapter Configurator dialog box.
2. Click Add to add a new part to the message. Select Attachment from the options in the enabled cell in the Format Plugin column.
3. Each plugin has a default MIME type, which is displayed in the Data Type column. The default data type for an Attachment format plug-in is a ZIP archive. If you know in advance what kind of file the attachment contains, click the cell in the Data Type column and enter the corresponding MIME type. For description of the data type used for each MIME type, refer to the Mime types topic.

The Attachment plugin does not require any more configuration at this stage, so no sub-tab for the Attachment part appears in the Message Body tab. However, when the dataslot is mapped, you must define an adapter input (example, named "Attachment") that can be mapped to a dataslot or a constant.

Mime types

The MIME type definitions help the email client to properly identify the type of the data contained in a message part, and to present this data in the proper format (if possible and supported by the client). If you select an incorrect MIME type for a given message part, the data may often be presented incorrectly by the client, although the data itself are likely not be corrupted.

Mime type names are specified in the following format:

type/subtype
where `type` is one of the top-level media types, as defined in RFC 2046, such as "text", "image", "audio", "video", and "application". The subtype further identifies the data format—for example "image/jpeg", "image/gif", "application/zip", "application/pdf", and so on.

Some of the most frequently used mime types are listed in the following table:

**Table 15: MIME types supported in the Email Adapter Configurator**

<table>
<thead>
<tr>
<th>MIME type</th>
<th>Data type format</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDF document</td>
<td>application/pdf</td>
</tr>
<tr>
<td>PostScript document</td>
<td>application/postscript</td>
</tr>
<tr>
<td>Rich text (RTF) document</td>
<td>application/rtf</td>
</tr>
<tr>
<td>ZIP archive</td>
<td>application/zip</td>
</tr>
<tr>
<td>GIF image</td>
<td>image/gif</td>
</tr>
<tr>
<td>JPEG image</td>
<td>image/jpeg</td>
</tr>
<tr>
<td>PNG image</td>
<td>image/png</td>
</tr>
<tr>
<td>TIFF image</td>
<td>image/tiff</td>
</tr>
<tr>
<td>HTML file</td>
<td>text/html</td>
</tr>
<tr>
<td>Plain text</td>
<td>text/plain</td>
</tr>
<tr>
<td>XML file</td>
<td>text/xml</td>
</tr>
</tbody>
</table>

For a full list of the recognized mime types, please refer to the documentation at the Internet Assigned Numbers Authority (IANA) website, found at the following URL address:

http://www.iana.org/assignments/media-types/

**Troubleshooting the Email Adapter**

The following issues may occur when you are using the Email Adapter.

- If you suspect the messages are not sent, please wait for a while before trying to debug the problem. Sometimes it takes a couple of minutes for an e-mail message to make its way through several mail exchanges to the recipient’s mailbox.

- To confirm that the adapter is properly configured, open the Adapter workstep’s Properties view and click the Configure button in the Configuration tab. Make sure that the Configuration tab in the Adapter Configurator includes the correct information. If this has been the problem, you can easily avoid similar mistakes in the future by preconfiguring your mail server information as a default. For more information, see the Preconfiguring the email adapter topic in Progress OpenEdge Business Process Modeler: User’s Guide.
• Check for errors in the log messages in the 
  OEBPS_HOME\ebmsapps\ApplicationName\logs\WorkstepName.log file and in 
  OEBPS_HOME\logs\bpserver.log or OEBPS_HOME\logs\bpmwebflow.log.

• You can increase the log-level in the 
  OEBPS_HOME\ebmsapps\ApplicationName\maps\WorkstepName\mapping.prop file. Open the 
  file with a text editor and change the value of the log-level parameter to a higher number (example, up 
  to 20). You have to re-deploy the maps for the process — either using the MapDeployer tool, or redeploying 
  the whole process. For information on the MapDeployer tool, check Deploying maps using MapDeployer 
  on page 37.

• If your e-mail server requires text in the messages to use LF (Linefeed) as the line separator instead of the 
  default CR+LF (Carriage Return/Linefeed), then you need to set the send.text.lf parameter to true 
  in the Extra Properties field on the Configuration tab of the Email Adapter Configurator.
File Managed Adapter

The File Adapter reads from and writes to a variety of file formats. It offers the benefits of:

- Support to popular file formats such as text files.
- Custom plug-in development for other file formats.

For details, see the following topics:

- File Adapter features
- How the File Adapter works
- Working with the File Adapter
- Writing custom format plug-ins

File Adapter features

The File Adapter features include:

- Supported file formats:
  - Text, for text files
  - Delimited, for delimited-text files (for example, comma-separated values)
  - XML, for extracting data from XML files or generating XML.
• Velocity Templates, for Text, XML, HTML, PostScript and other formats, all generated with the Velocity template engine.

**Note:** The text files generated by the File Adapter use the 0x0A end-of-line character. Certain DOS and older Windows programs and utilities may not recognize the new line character, since they may require the 0x0D;0x0A character sequence for an end-of-line separator. Make sure you use the appropriate utility to manipulate such files. For example, if you encounter problems using the Notepad editor, use Wordpad to edit or view text files generated by the File Adapter.

• Custom formats allowed with a plug-in development kit as described in Writing custom format plug-ins topic in *Progress OpenEdge Business Process Modeler: User’s Guide*.

• The File Adapter uses the default MapConfigurator GUI as described in *Understanding the Managed Adapter framework* on page 19

---

**How the File Adapter works**

The following diagram illustrates the architecture of the File Adapter and the Format Plug-ins.

**Figure 5: File Adapter architecture**

The File Adapter consists of Design-time components and Run-time components. The Design-time components are used when you define a business process and the Run-time components are invoked when the File Adapter is executed.
Design-time components

When you design your Business Process Server process with Progress Developer Studio for OpenEdge, the configuration parameters of the File Adapter are defined using the File Adapter Configurator. These parameters include:

- File name and path.
- Mode — read, write, or append.
- File format — Text, Delimited-text, XML, Velocity template, or custom format.

After you specify the file format, the File Adapter configurator verifies if the given format requires any additional configuration. In case additional configuration is required, the configurator loads and displays the corresponding File Configuration Plug-in GUI.

Some file formats do not require any additional user-defined configuration. For example, plain text files can be read, written or appended to, without the need for File Adapter to know anything about the meaning of the data in the file. In such cases, the plug-in does not have its own format-specific GUI. On the other hand, extracting data from an XML file requires the XPath statement to the specific field to be configured, and the XML plug-in has a GUI configurator that allows browsing the XML template and entering XPath expressions.

Run-time components

When the File Adapter is executed, it receives the mapped input dataslots from the Business Process Server process, as well as the configuration defined at design time. While the File Adapter is capable of finding and opening the file, it does not try to process the file data, but delegates this task to the corresponding run-time Format Plug-in.

Every Format Plug-in has a run-time component that processes the file data. The run-time Format Plug-in receives the configuration in the same form it was defined by the configurator Format Plug-in GUI. The run-time Format Plug-in performs the required data manipulations and passes the result back to Business Process Server.

Working with the File Adapter

The File Adapter is a Managed Adapter and its usage is standardized along the same lines as the other Managed Adapters. For more information, see the adapter configuration and mapping framework described in Understanding the Managed Adapter framework on page 19

Configuring the File Adapter

You can configure the File Adapter in your process by performing the following procedures.

1. Click the Assign Participants link in the Tasks pane. From Adapters > Managed > FileAdapters, drag the File Adapter to the Content pane.

2. Double-click the File Adapter workstep to open its Properties view.

3. From the Properties view for the workstep, click the Configuration tab. This tab includes the Configure, Change mapping, and Advanced properties buttons for defining the adapter configuration and mapping.

4. Click Configure to open the File Adapter Configurator dialog box.
You can enter the basic configuration for the File Adapter:

- **Mode.** Defines the File Adapter behavior - whether it should Read a file, Write a file (overwriting the previous data, if any), or Append to an already existing file.

- **FileName** and **Path Prefix.** These fields allow you to specify the full path to the file you are going to process. You can enter values for either of them, or you can leave the fields empty. If these fields are empty, you are allowed to map any of them to a dataslot later.

**Note:** Generally, the **FileName** field stores just the name of the file, while the directory path is given in the **Path Prefix** field. This method allows you to dynamically map the file name from a process dataslot, while the directory path remains constant.

- **Data Format.** Specifies the format of the file. Options include **Text** (a simple text file), **Velocity Template** (complex text using the Velocity Template engine syntax), Delimited (a comma-separated list of values), or **XML** (an XML file). You can select one of the available plug-ins, including any custom Format Plug-ins you may have developed.

**Note:** For more information on data types supported by the Managed Adapter Framework, see the Supported data types section in the Progress OpenEdge Business Process Server: Managed Adapters Guide.

Once you select the plug-in, the **File Format** tab is enabled to display the corresponding plug-in GUI configurator.

Note that all plug-ins may not support all three modes - Read, Write, and Append. If you select a mode that is not acceptable for the given plug-in, you get a warning message and the mode is switched to the default plug-in mode.

The next sections deal with the configuration of each of the default plug-ins.

### Using the Text Format plug-in

The Text Format plug-in can be used for the Email Managed Adapter and the File Managed Adapter.

**To configure and map the Text Format plug-in:**

1. For a File Managed Adapter, open the **File Adapter Configurator** dialog box. Select the **Mode** as Write or Append, and enter the file name, preferably pointing to your temporary directory. The Text Format plug-in does not have any extra configuration and the format-specific tab (**File Format** tab) is not enabled.

   For an Email Managed Adapter, open the **Email Adapter Configurator** dialog box (refer to Configuring the Email Adapter on page 66). Click **Add** to add a new part to the message. Select **Text** from the options in the enabled cell in the Format Plug-in column.

2. Click **OK**. The **Dataslot Mapping** box is displayed.

   Alternately, from the **Configuration** tab in the **Properties** view, click **Change mapping** to open the **Dataslot Mapping** dialog box.
3. In the **Source** drop-down list, select a dataslot where your text should come from. The value of this dataslot will be stored in your output file. Note that depending on the File Adapter configuration, the mapper window may look different or may contain additional fields.

4. Click **OK**. The File Adapter is now fully configured for text files.

### Using the Velocity Template plug-in

The Velocity Template plug-in can be used to generate complex text using the Velocity template engine syntax. The templates can contain loops, if-then-else blocks, and other expressions defined in the Velocity Template Language (VTL). For more information on Velocity and VTL, refer to the Velocity documentation, available at the following URL address: [http://velocity.apache.org/](http://velocity.apache.org/)

A typical use of the Velocity plug-in is to substitute variables in a template. The syntax used by Velocity for a variable is `$$VAR` or `${VAR}`. For example, if you want to use the Velocity plug-in to generate a message to a user, you can define the template as follows:

```
Hello ${USER}: Your account was activated on ${DATE}.
```

You can click the **Extract Variables** button (in the **Velocity Template** tab of the **File Adapter Configurator** dialog box) to display a list of variables used in the template. If necessary, you can edit this list. Each variable will then be represented as an adapter input, available for mapping to a dataslot, or as a constant.

You can use the Velocity Template plugin for File Managed Adapters and Email Managed Adapters:

#### To configure Velocity Template plug-in:

1. For a File Managed Adapter, open the **File Adapter Configurator** dialog box, select the **Mode** as Write or Append and enter the file name. Select the **Data Format** as Velocity Template. After you have selected Velocity Template, the generic **File Format** tab is enabled and renamed to a format-specific tab called **Velocity Template**.

   For an Email Managed Adapter, open the **Email Adapter Configurator** dialog box (refer to **Configuring the Email Adapter** on page 66). Click **Add** to add a new part to the message. Select **Velocity Template** from the options in the enabled cell in the Format Plugin column. Open the **Message Body** tab.

2. Click the **Velocity Template** tab. The **File Adapter Configurator** dialog box for the Velocity Template plug-in appears:

3. In the top text area, enter the text of the template. Refer to the Velocity documentation for help on VTL. If you already have a file with the prepared template, you can load it by clicking **Load Template**.

   This template generates a file in the following format:

   ```
   Product list for <CompanyName>:
       --> <Product1>
       --> <Product2>
       --> ...
   ```

   We recommend using the `${varname}` notation for the Velocity plugin. The company name and the list of products are defined in the variables `${company}` and `${products}` respectively. The `${company}` is a scalar variable, while `${products}` is a list.

4. The Velocity Template plug-in needs a list of variables, whose values will be set from dataslots. You can either enter the names of all such variables in the bottom text area, or click **Extract Variables** to build such a list automatically. In our example, clicking **Extract Variables** generates the list of variables shown below:

   The Velocity plug-in configurator listed all the variables present in the template. Note, however, that the `${product}` variable is used only to iterate through the `${products}` list, so it should not receive its value from a dataslot as an adapter input. Edit the variable list, removing the `${product}` variable.
Depending on the complexity of your template and the types of variables you are using, you may need to add or remove more variables.

5. The configuration of the Velocity Template plug-in is now complete. Click OK to close the File Adapter Configurator, which also opens the Dataslot Mapping dialog box for the File Adapter workstep.

You have one input parameter for each Velocity variable you listed. Map the "company" input to a CHARACTER dataslot, containing the company name (@CompanyName). Map the "products" input to a LIST dataslot, containing the list of products for the given company (@CompanyProducts).

This process is ready to be published and executed. A sample output of the Velocity Template plug-in is shown below:

```
Product list for Fruit Co:
  --> Apples
  --> Oranges
  --> Peaches
```

You can use the Velocity Template plug-in to also generate XML, HTML, Postscript, and other files. The example above shows only a small part of the Velocity template engine functionality. For more information on Velocity and the VTL syntax, refer to the Velocity documentation.

### Using the Delimited Text plug-in

The Delimited Text plug-in provides support for files containing a sequence of records, each one a comma-separated list of values. Although comma is the default, you may select any other string (except new line) as field separator. The record separator is always the new line character.

**Note:** The separator used in the Delimited Text plug-in must not be used as a part of the text.

**To configure Delimited text plug-in:**

1. In the File Adapter Configurator dialog box, select the Mode as Read, Write or Append, and enter the file name - preferably pointing to your temporary directory.

2. Select Data Format as Delimited. After you have selected Delimited, the generic File Format tab is enabled and renamed to a format-specific tab called Delimited.

3. Click the Delimited tab.
   
   In this tab you can enter the name and type of each field in your record.

4. Click Add to define a new record. The Parameter Editor dialog box appears.
   
   a) Define a Name for the field.
   
   b) Define the Type of the value it will contain.
   
   c) Click OK to return to the configurator box.

5. Continue adding more fields as appropriate.

6. Click Modify to modify the field definition.
   
   Click Remove to delete a field.

   Click Move Up or Move Down to move the field to match the desired record structure.

7. Click Advanced to define a field separator other than the default (","). The Advanced Parameters dialog box is displayed.
a) Select the **Separator String** from the drop-down list or enter your custom character/string.  

For Write or Append mode, the separator is simply a string that will be inserted between the fields. The only special characters are "space" and "tab" that are presented as \s and \t in the pull-down list, so they can be distinguished visually.  

In Read mode, the separator is not a simple string, but a regular expression (regex) that will be used to split the record into fields. You should be aware of this, especially if the separator you are using has a special regex meaning. For example, if you want the fields to be separated with ",", in Read mode you should define the separator regex as \*.

Using regular expressions to split the fields allows for more complex separators to be defined. For example, if the fields in your file are delimited with one or more blank (space or tab) characters, you can define the separator as \s+. For additional information on regular expressions syntax, check the API documentation for `java.util.regex.Pattern`, available online from Sun Microsystems.

b) The **LineNumber** field is enabled when you set the mode to Read. In this case, if the delimited-text file you are reading has more than one record, you can specify which record you want to read. The default is to read the first record (1). You can enter any record number you want, or select **Last** for the last record of the file.  

You can also select **Mapped** indicating that the record number should come from a dataslot. In the last case, the record number field will be visible when you define the mapping for the adapter, allowing you to map the value from the dataslot, containing the record number.

c) Click **OK** to return to the Configurator dialog box.

8. Click **OK** to return to the Properties view.

9. Click **Change mapping** to open the **Dataslot Mapping** dialog box. Open the Data tab.

10. For Write or Append mode, select a dataslot under **Source** where the fields in your record should come from. The value of this dataslot will be stored in your output file. For Read mode, select a dataslot in the **Target** drop-down list to which the fields in your record should store. Note that depending on the File Adapter configuration, the mapper window may look different or may contain additional fields.

11. Click **OK**. The File Adapter is now fully configured for delimited text files.

**Using the XML Format plug-in**

The XML Format plug-in allows you to extract data from XML files, or to generate XML data from the BP Server dataslots.

You can use the XML plugin for File Managed Adapters and Email Managed Adapters.

**To configure XML format plug-in:**

1. For a File Managed Adapter, open the **File Adapter Configurator** dialog box, select the **Mode** as Read or Write and enter the file name. Select the **XML** option for the Data Format. After you have selected XML, the generic **File Format** tab is enabled and renamed to a format-specific **XML** tab.

   For an Email Managed Adapter, open the **Email Adapter Configurator** dialog box (refer to Configuring the **Email Adapter** on page 66). Click **Add** to add a new part to the message. Select **XML** from the options in the enabled cell in the Format Plugin column.

2. Open the **XML** tab.

   In this tab you can enter the Name, XPath, and Type for each field in your record. Each of the parameters defined here corresponds to a node in the XML file. The location of each node in the XML tree is specified by an XPath expression.
At this step, you need to define a template XML as explained in the section Defining the template XML in Progress OpenEdge Business Process Modeler: User's Guide.

3. Click **Template** to open the **Select XML Template** dialog box.
   a) Select the template file. If you do not want to use the pre-configured template, or if your XML template comes from a dataslot, click **Clear** to clear the file name.
   b) Click **OK** to return to the **File Adapter Configurator**.

4. Click **Add** to define a new record. The **Parameter Editor** dialog box appears.
   a) Define a **Name** for the field.
   b) Define the **Type** of the value it will contain.
   c) Enter the XPath expression identifying the node directly, or click in the XPath field to open the **XPath Browser** dialog box.
   d) Select the node. The XPath expression is displayed in the text field at the bottom. If required, you can manually edit this expression. For example, if you are looking for a particular record, one of a sequence of records, in an XML file - in this case you may specify the record number or search condition. For more information, refer to the Using variables and conditions in XPath topic in Progress OpenEdge Business Process Modeler: User's Guide.
   e) Click **OK** to return to the **Parameter Editor** dialog box.
   f) Click **OK** to return to the **File Adapter Configurator**.

5. Click **Modify** to modify the field definition.
   Click **Remove** to delete a field.
   Click **Up** or **Down** to move the field to match the desired record structure.

6. Click **OK** to open the **Dataslot Mapping** dialog box.
   Alternatively, you can open the **Dataslot Mapping** dialog box by clicking **Change mapping** in the **Properties** view.

7. Under **Source**, select a dataslot where the fields in your record should come from. The value of this dataslot will be stored in your output file. Note that depending on the File Adapter configuration, the mapper window may look different or may contain additional fields.
   For more complex mapping of XML files, refer to the section Using variables and conditions in XPath in Progress OpenEdge Business Process Modeler: User's Guide.

8. Click **OK**. The File Adapter is now fully configured for XML files.

### Defining the template XML

Consider that you need to generate an XML file using values from dataslots in your process and the XML needs to be generated in the following format:

```xml
<library>
  <book id="187510">
    <author>Johnny Appleseed</author>
    <title>Pocket Guide to Apple Trees</title>
  </book>
</library>
```
You need to first create a template XML. Using a text editor, create the library.xml template, with the following content:

```xml
<library>
  <book id="0">
    <author></author>
    <title></title>
  </book>
</library>
```

Note that the author’s name and the book title are removed from the example. While not strictly necessary, this will ensure that if these fields are not mapped to any dataslots, they will be empty and not contain any incorrect data. You can also enter default strings here such as "unknown".

Once you save the template file, you can return to the Process Designer.

**Note:** If your XML template refers to a DTD file, make sure that the DTD is specified in such a way that it is available for the XML parser when the adapter is invoked by the application server. For the DTD file name, either use global URL, or at least fully qualified file name that will be available to the Business Process Server.

---

### Using variables and conditions in XPath

When the XML file is more complex, the additional information can be given in the XPath expression. For example:

```xml
<library>
  <book id="187510">
    <author>Johnny Appleseed</author>
    <title>Pocket Guide to Apple Trees</title>
  </book>
  <book id="187621">
    <author>Johnny Appleseed</author>
    <title>Another Pocket Guide to Apple Trees</title>
  </book>
  <book id="273531">
    <author>Johnny Appleseed</author>
    <title>Apples for Dummies</title>
  </book>
</library>
```

In this case, if you need to get data from the first record only, you can define the XPath expressions for selecting the author’s name and the title for the book in the first record as shown below:

```xml
/library/book[1]/author
/library/book[1]/title
```

This definition will always return the same (first) record. If you want to make the record number configurable at run time, you can change the expressions as follows:

```xml
/library/book[${RecNo}]/author
/library/book[${RecNo}]/title
```

Note that you have a new parameter named "RecNo" that could be later mapped to a dataslot. As you are reading the file, the AuthorName and BookTitle are output fields (returned from the File Adapter), while the new parameter RecNo is input (passed to the File Adapter from the dataslot, before the adapter is executed).
More often, instead of using a record number to locate your book, it is more convenient to select the record by the ID attribute. If you want to search by an ID that comes with a dataslot, you can define the XPath expressions as follows:

```
/library/book[@id='${BookID}']/author
/library/book[@id='${BookID}']/title/library/
```

This will generate a new adapter input, called "BookID" that can then be mapped to a dataslot. Here you can put the book ID in a dataslot, and then invoke the File Adapter. The adapter will search the XML file for a book with this specific ID number and will return the author's name and book title.

**Writing custom format plug-ins**

If you want to use a data format that is not covered by the default plug-ins, you can develop custom plug-ins suited to your requirements. This section guides you through the step-by-step process of the custom plug-in development.

As an example, we will develop a simple plug-in for logging messages in a file. The plug-in stamps the current date and time for each message logged, and allows some level of customization of the message format.

**Objective**

The custom Format plug-in supports a combination of the three modes - Read, Write, or Append.

The Read mode is where a plug-in is presented with an open stream to the file data, and is expected to produce a number of output parameters (outputs) from the file data. Each of these outputs can be mapped to a process dataslot. The master file adapter takes care of locating the file and opening the input stream.

From the plug-in point of view, the Write and Append mode are the same — the plug-in is presented with an open output stream to the file, and a set of input parameters (inputs). Its task is to generate data in whatever format the plug-in was designed to handle, and to write this data to the output stream. Locating and opening the output file in the correct mode, as well as supplying the input parameters, is done by the File Adapter and not by the plug-in.

Apart from the input and output data, and the input or output stream, the plug-ins are also passed a custom configuration object. This plug-in specific configuration contains any extra information that is needed by the plug-in to function.

The plug-in can be graphically represented by the following figure.

**Figure 6: File Adapter - Custom format plug-in**

![Diagram of File Adapter - Custom format plug-in]

Note that the diagram represents only the Write/Append mode.

The message, or input parameter, to be logged is received from Business Process Server. The format of the log record is the custom configuration. The plug-in formats the log message according to the configuration and writes it to the provided output stream.
Plug-in definition

The file `OEBPS_HOME/conf/resources/adapters/formatplugins.properties` contains a list of all available plug-ins. For each plug-in, several properties are defined, including the plug-in name to be displayed when selecting the plug-in from the list in the File Adapter configurator, plug-in configurator class, run-time class for data processing, list of available modes, and MIME-type of the plug-in data.

First, decide the modes (Read, Write, Append) supported by the plug-in. The sample plug-in is intended for logging, and hence only the Write / Append mode is required. The most used mode would evidently be "Append" (adding each message to the end of the log-file), but "Write" (rewriting the file to contain only the most current message) also can be useful.

The first step is to define the plug-in and the supported modes in the file `OEBPS_HOME/conf/resources/adapters/formatplugins.properties`. Open the file in a text editor and add `log` to the `enabled` plug-ins, as shown in the list at the top of the page:

```
enabled=txt,velocity,csv,xml,log
```

Next, enter the following definition for the new plug-in:

```
# Log format plugin
name.log=Log
log.runtime=com.savvion.sbm.adapters.file.plugins.Log
log.config=com.savvion.sbm.adapters.file.plugins.LogConfig
log.modes=AW
log.mtype=text/plain
```

Each plug-in in the `formatplugins.properties` file is defined by an identifier — in our case the identifier is `log`. The other properties are:

- `name.log` sets the display name of the plug-in to "Log".
- `log.runtime` contains the class name of the plug-in at run time.
- `log.config` contains the class names of the plug-in configurator class.
- `log.modes` states that the plug-in supports the A (Append) and W (Write) operations.
- `log.mtype` defines the MIME-type of the plug-in as plain text.

**Note:** The first mode in the modes property is the default mode for the plug-in. If the user tries to set the plug-in mode to an invalid value, a warning message will appear and the default mode will be automatically selected. If the modes property is not specified, "RWA" is assumed.

After defining the custom plug-in, save the changes and restart Progress Developer Studio for OpenEdge (or Business Process Modeler).

Plug-in Configurator

Apart from the configuration of the File Adapter (file name, directory, mode), each plug-in may have its own configuration, used to tune the way it processes the data. Not all plug-ins need custom configuration. For example, the Text plug-in only writes the string it is given, and no configuration is needed.
One of the requirements for the custom Log Format plug-in we are creating is to provide some level of customization of the format of the messages stored in the log file. This can be achieved by providing a format string that will match each record stored in the file, such as:

[D] M
D - M
M [D]

where M is the message, and D is the date/time when this message was logged.

The plug-in configuration is maintained by the File Adapter and can be any serializable object. However, using a String object for plug-in configuration has some advantages and is highly recommended. If the configuration of your plug-in is relatively simple, we recommend that you devise some mechanism to store it as a string. If the configuration is more complex, we recommend that you define a configuration wrapper object that can produce XML string, and can be initialized from XML string. This will make the configuration files more readable, and will also provide some performance advantages over simply using serializable objects.

In case of the Log Format Plug-in, the configuration can consist simply of the format string.

Every Format Plug-in must have its own configurator class. The configurator class may or may not have a GUI for configuration, but the configurator class is mandatory. If the configurator class extends the java.swing.JDialog class, the GUI is enabled; otherwise, the GUI is disabled.

The configurator class must implement the FormatConfigPlugin interface, as described below:

```java
public interface com.savvion.sbm.adapters.file.FormatConfigPlugin {
    public void setMode(int mode) throws Exception;
    public void setFormatConfig(Object formatConfig) throws Exception;
    public Object getFormatConfig() throws Exception;
    public Vector getParameters() throws Exception;
}
```

The setMode() method is called at design time when the user selects an adapter mode, or at runtime, when the File Adapter is invoked. The values passed to setMode() are one of the following constants, defined in the com.savvion.sbm.adapters.file.FileAdapterConfig class:

- MODE_READ
- MODE_WRITE
- MODE_APPEND

The setFormatConfig() method is executed at design time or at runtime, when the format configurator is initialized. The formatConfig parameter is the custom plug-in configuration object, which is the message format string for the Log Plug-in.

The getFormatConfig() method is used only at design time, to generate the custom format configuration object. In case there is a GUI associated with adapter configurator, the formatConfig will be generated from the values of the GUI controls set by the user. In case the File Adapter does not require GUI configuration, this method may return "null", or any static configuration object.

The getParameters() method is the most important method. It must return a Vector of the com.savvion.sbm.adapters.file.Parameter objects, each one representing an adapter input or output parameter that defines the set of values your File Adapter accepts as input, and the set of values it generates as output. For complete information on the Parameter class, refer to APIs for custom format plug-in section in the Progress OpenEdge Business Process Server: Managed Adapters Guide.

The source of the custom File Adapter configurator is described below:

```java
package com.savvion.sbm.adapters.file.plugins;
import com.savvion.sbm.adapters.file.*;
import java.util.*;
import java.awt.*;
import javax.swing.*;
public class LogConfig extends JPanel
```
implements com.savvion.sbm.adapters.file.FormatConfigPlugin
{
    JTextField formatField = new JTextField();
    
    public LogConfig()
    {
        this.setLayout(new GridBagLayout());
        GridBagConstraints c = new GridBagConstraints(); c.gridy=0;
        c.gridx=0; this.add(new JLabel("Message format:"),c);
        c.gridx=1; this.add(formatField,c);
        formatField.setColumns(10);
    }
    
    public void setMode(int mode) {}
    
    public void setFormatConfig(Object formatConfig) {
        if (formatConfig instanceof String) {
            formatField.setText((String)formatConfig);
        } else {
            formatField.setText("[D] M"); // default format
        }
    }
    
    public Object getFormatConfig() {
        return formatField.getText();
    }
    
    public Vector getParameters() {
        Vector rtn = new Vector();
        rtn.add(new Parameter("MSG", // parameter name
            null, // access (use mode default)
            "Message", // parameter title
            "java.lang.String", // parameter type
            null, // default value
            "true" // mandatory
        ));
        return rtn;
    }
}

Note that the File Adapter Configurator extends from the class javax.swing.JPanel, indicating that the plug-in has a GUI interface where the user can define the log message format. The GUI interface defined in the LogConfig() constructor is shown in the following figure.

**Figure 7: File Adapter - Custom format plug-in GUI**

![Message format: [D] M](image)

The user enters the format string in this GUI.

The setMode() method is not used for the Log Format Plug-in, since the adapter is expected to work in either Write or Append mode, and not in any single default mode.

In the setFormatConfig() method, the text field is initialized with the message format string. In case of invalid values, the format string is set to the default format [D]M.

The getFormatConfig() method simply returns the format string entered by the user in the text field.
The `getParameters()` method is used to return the list of input and output parameters for the plug-in. The Log Plug-in receives one single string parameter — the message to be logged, so in the `getParameters()` method one single parameter is returned, named "MSG". The "access" field of this parameter can be either "I", "O", or "IO" (Input, Output, and Input/Output parameter), or it can be "null", which will use the default for the given adapter mode.

**Note:** Giving "null" for a parameter access, means that the access will be determined by the FileAdapter configurator based on the selected adapter mode. For example, if the mode is Write or Append, the default access would be "I", because in this mode the File Adapter is expected to receive some input values from your Business Process Server installation and store them in file. For the Read mode, the default value will be "0", because the content of the file is expected to be presented to Business Process Server as a set of outputs for the adapter parameters. You can use "null" access in most cases when writing custom format configurators.

The parameter title, "Message" is for display purposes, and will be used when defining dataslot mapping. The parameter type is the fully qualified Java class type that you expect to receive from Business Process Server. For the list of supported types, see the "Supported Data Types" section in the "Custom Managed Adapters" chapter in the Customization Guide.

The "mandatory" attribute is set to either "true" or "false" to denote whether or not the given parameter is essential for the work of the plug-in. In our case, we must have the message string in order to log anything in the log file, so we set "mandatory" to "true". In other situations, when the plug-in may work successfully with incomplete data, you can set this attribute to "false". The difference between mandatory and non-mandatory parameters will appear when the user defines the dataslot mapping — if a mandatory parameter is left unmapped, a warning message will pop-up.

**Note:** For a number of reasons, the "mandatory" attribute has effect only on input and input/output parameters.

When one of your business processes using the Log Format Plug-in is configured, the format configurator will be embedded in the File Adapter Configurator GUI.

### Writing the Plug-in run time

For writing the Plug-in Run time, the class must implement the interface `com.savvion.sbm.adapters.file.FormatPlugin`

```java
public interface FormatPlugin {
    public void write(Hashtable input, OutputStream os, Object formatConfig) throws Exception;
    public Hashtable read(Hashtable input, InputStream is, Object formatConfig) throws Exception;
}
```

The `write()` method is used in both the Write and Append modes. Here, the "input" hashtable contains all the adapter inputs, defined by the plug-in Configurator's `getParameters()` method. In our case, it contains only the "MSG" parameter - the message to be logged in the file. The "os" output stream is already opened by the File Adapter, so the plug-in has to just write the message in the correct format to this stream. The `formatConfig` object is the custom plug-in configuration. In our case, this is the message format string ([D]M). The `write()` method can throw an exception if something goes wrong. This exception is logged in the adapter logfile and the workstep execution is suspended.

The `read()` method accepts a similar set of parameters and is used in the Read mode. The "input" Hashtable contains any optional input parameters. The "is" input stream is already opened from the source file. The "formatConfig" object is the optional custom configuration for the plug-in. The `read()` method is expected to return a Hashtable, containing all the outputs, as defined by the plug-in configurator's `getParameters()` method.
Note: Usually the "input" Hashtable parameter of the "read()" method is empty, because in Read mode the plug-in does not require any inputs - in this mode it reads data from the file and returns the processed output. However, some plug-ins with more complex configuration may still require to be supplied with additional inputs, even for this mode - usually to fine tune the way the File Adapter is executed. Those inputs can be mapped to dataslots, and provide a way to control the plug-in behavior dynamically.

The Log Plug-in will support only Write and Append modes, so the only method to implement is write(). The source of the Log Plug-in Run time is described below.

```java
package com.savvion.sbm.adapters.file.plugins;
import com.savvion.sbm.adapters.file.*;
import java.io.*;
import java.util.*;

public void write(Hashtable input, OutputStream os, Object formatConfig)
    throws Exception
{
    // Prepare format string, message, and date
    String line = (String)formatConfig;
    String msg = input.get("MSG");
    String date = new java.util.Date().toString();

    // Prepare message
    line=line.replaceAll("%D",date);
    line=line.replaceAll("%M",msg);
    line=line+"\n";

    // Write message in the log
    os.write(line.getBytes());
}

public Hashtable read(Hashtable input, InputStream is, Object formatConfig)
    throws Exception
{
    return null; // not supported
}

public class Log implements FormatPlugin {
}
```

In the write() method, we first obtain the message format ("D" M, or any custom value) - this is our configuration object. Next, we load the message itself, as the "MSG" element of the input Hashtable, as well as get the current date and time.

After this, we prepare the message according to the custom format, and write the message to the log.

You can leave the read() method empty, since this mode is not supported.

Installing and running the plug-in

After compiling the Format Plug-in, pack the classes in a jar file and include them in the classpath, as well as in the application server system classpath. After restarting the application server, you can start using the new Plug-in.

A sample Business Process Server application with the Log Format Plug-in is shown in the following figure.

Figure 8: File Adapter - Log Format plug-in an application
The SetMessage human workstep is used to enter the text of the message to be logged. The LogMessage adapter workstep invokes the Log Plug-in of the File Adapter to store the message in the logfile.

To install and run the plug-in:

1. Define a CHARACTER dataslot, MessageDS. This dataslot will contain the message text.
2. Double-click the SetMessage workstep and add this dataslot to those listed on the Dataslots sub-tab.
3. Double-click the LogMessage workstep and verify that GenericFileAdapter is the performer listed in the Performer field.
4. In the Configuration tab, click Configure to display the File Adapter Configurator dialog box.
5. Select the Mode as Append.
6. Select the Data Format as Log, and configure the file name and prefix, as shown above. Ensure that the directory exists on the Business Process Server, and you have permissions to write to the output file you selected.
7. Click the Log tab. This tab contains the designed custom format configurator.
   You can modify the default message format ([D] M) if required.
8. Click OK. The File Adapter is now configured.
9. The Dataslot Mapping dialog box appears, as shown in the following figure.
   Alternatively, you can open the Dataslot Mapping dialog box by clicking Change mapping in the Configuration tab in the Properties view for the LogMessage workstep.
   a) Map the MessageDS dataslot to the Message input parameter.
   b) Click OK.

Your business process is now ready to be published. Publish and run the process. Verify that the message you entered appears in the logfile and is in the correct format.

Troubleshooting the custom format plug-in

- The Log Plug-in is not visible in the File Adapter Configurator.
  Ensure you exit and then re-open Progress Developer Studio for OpenEdge.
  If you do not see the Log Plug-in name in the Data Format combo box, the plug-in may not be properly defined in the formatplugins.properties file. Go to the OEBPS_HOME\conf\resources\adapters directory, open this file with a text editor, and enter the plug-in definition as described in the Plug-in definition on page 87.

- Error when the Log Plug-in is selected in the File Adapter Configuration.
  Ensure that the plug-in configurator class is included in the product classpath.

- Classpath error when executing your business process.
  Ensure that the plug-in run time class is included in the system classpath of the application server where Business Process Server is running. You may need to restart the application server after changing the system classpath.

- Error accessing the log-file when executing the adapter.
  Check if the file name and path are correct, and that you have write permission. Use fully qualified path when configuring the File Adapter.
In case of any other problem, check the log files in the OEBPS_HOME\logs directory. Error messages in this file can help you to identify the problem.
FTP Managed Adapter

The FTP Managed Adapter allows you to transfer files between an FTP server, a local file system, and Document dataslots in your business processes.

For details, see the following topics:

- FTP Adapter features
- Working with the FTP Adapter
- Troubleshooting the FTP Adapter

FTP Adapter features

The FTP Adapter features include:

- Transfer of files from and to FTP servers and local file system and Document dataslots.
- Support of FTP transfer in TEXT/BINARY mode.
- Support of Active/Passive FTP.
- Conformance of the FTP Adapter configuration and mapping to the specifications of our Managed Adapter Configuration and Mapping Framework (see Understanding the Managed Adapter framework on page 19).
Working with the FTP Adapter

The FTP Adapter is a managed adapter and its usage and framework is standardized along the same lines as the other Managed Adapters provided with the Business Process Server installation.

Configuring the FTP Adapter

You can assign and configure the FTP Adapter to an Adapter workstep in your process by performing the following procedures.

To configure the FTP adapter:

1. Click the Assign Participants link in the Tasks pane. Expand Adapters > Managed > FTP Adapter folder and drag the default GenericFTPAdapter (or a previously configured FTP adapter instance) from the Tasks pane to the Content pane, creating an FTPAdapter workstep.
2. Double-click the workstep to open its Properties view, and click the Configuration tab. This tab includes the Configure and Change mapping buttons for defining the adapter configuration and mapping respectively.
3. Click Configure to open the configurator box of the FTP Adapter Configurator.

Note: You can also open the FTP Adapter Configurator using the new Managed Adapter Browser (available only from Progress Developer Studio for OpenEdge) functionality. In addition to configuring, you can use the Managed Adapter Browser to perform functions like creating a copy, renaming, deleting, as well as importing and exporting the configuration information. For more information, refer to the "Using the Managed Adapter Browser" section of the OpenEdge Getting Started: Developing BPM Applications with Developer Studio.

4. The FTP Adapter Configurator dialog box includes two tabs, Source Location and Target Location. Use the tabs to specify the Source and Target locations, respectively. The FTP Adapter Configurator also includes the following three file location options:

- **Document dataslot.** The file is downloaded from (or uploaded to) a Document dataslot in your business process.
- **Local Filesystem.** The file is loaded from (or saved to) a location on the Business Process Server's file system. For more information, see Accessing files on the Local FileSystem on page 96
- **FTP server.** The file is downloaded from (or uploaded to) a remote FTP server. For more information, see Accessing files on a remote FTP server on page 97.

Any combination of source and target locations is allowed.

Accessing files on the Local FileSystem

You can configure the FTP Adapter to load or save a file on the local (Business Process Server) file system.

To access files on the local filesystem:
1. Select the **Local Filesystem** option in the **Source Location** or **Target Location** tab.

2. You can now enter the Path Prefix and File Name. The values of those parameters will be concatenated to specify the full path to your file. For example, entering `C:\temp` for Path Prefix, and `sample.txt` in File Name will point to the `C:\temp\sample.txt` file. You can also leave either (or both) of these parameters empty.

3. When you click **OK**, the **Dataslot Mapping** dialog box appears. If you have filled out the Path Prefix and File Name fields, the dialog box does not display the **Source** tab as the source location has been defined.

4. Assuming that only the Path Prefix has been entered at configuration file, the **Dataslot Mapping** dialog box appears.

5. Map the `dsxml` dataslot to provide the name of the file. With the path prefix defined in the FTP Adapter Configurator, the full path of the file would be: `C:\temp\@dsxml`, where `@dsxml` is the value of the `dsxml` dataslot.

6. Click **OK** to complete the mapping. If you have left the File Name field blank, a Warning appears that informs you to provide a File Name. If you have left both fields blank, you are warned to provide values for both missing parameters. The parameter value can be a constant (specified directly in the Map Configurator dialog), or may come from a CHARACTER dataslot.

When using the FTP Adapter to store a file on the local filesystem, the target file name is not mandatory. If you do not specify a name for the target file, the file name would be the same as the source file.

**Accessing files on a remote FTP server**

You can map an input or output to a file located on a remote FTP server.

**To access files on a remote FTP server:**

1. Select the **FTP Server** option in either the **Source Location** or **Target Location** tab to map the corresponding input or output to a file located on a remote FTP server.

2. Specify the FTP Server’s host name, optionally the port name (if different than the default, 21), the Username and Password, the complete path of the directory and the name of the file located in the remote Business Process Server. If you want to use the default FTP directory, enter "." for the **Directory** parameter.

3. Additionally, you can select the **ASCII Mode** check box, if you want your file to be transferred as a text document, or the **Passive** checkbox if you want to connect to the FTP server in Passive mode (for instance, in order to bypass a firewall).

4. Click **OK** in the **Dataslot Mapping** dialog box that appears to complete the mapping. If you do not supply any of the server parameters in the FTP Adapter Configurator, a prompt to map the missing values to dataslots appears.

   Again, if you do not provide a file name for a target FTP destination, the file name will be the same as the name of the source file. Note, however, that the Username, Password and Server name are required in all cases.

**Troubleshooting the FTP Adapter**

If you encounter problems when using the FTP Adapter, you need to check the following points:

- When transferring data from or to Document dataslots, make sure the dataslots were selected properly in the map configurator. Note that files can be downloaded only from single-document dataslots.
• If you encounter problems creating or accessing a file on the local file system, make sure that you have properly specified the file name and path, and that the path and file are available at the computer, where the Business Process Server is running. Make sure that the directory and file you are trying to access have permissions allowing the server to read or write to it.

• If you cannot upload or download from an FTP Server, confirm that the server/port/username/password parameters you specified are correct. Try connecting to the FTP server, and make sure that the directory and file you specified are present at the FTP Server and with the appropriate access permissions. If you have problems establishing the FTP connection, you can try selecting the Passive check box in order to connect in the passive mode.

• If the preceding tips do not help, check the FTP Adapter log file, and the BP Server/BPM Webflow log files. For BP Server applications, the FTP Adapter log file is located in the <OEBPS_HOME>\ebmsapps\<APPLICATION_NAME>\log directory; for Web (BPM Webflow) applications, the log file is in the <OEBPS_HOME>\bpmwebflow\<APPLICATION_HOME>\log directory. The bpserver.log and bpmworkflow.log files are located in the OEBPS_HOME\logs directory.
JMS Managed Adapter

The JMS Adapter enables you to use Java Message Service (JMS) to communicate asynchronously with external systems and facilitates data exchange between your business processes and external applications.

For details, see the following topics:

- JMS Adapter features
- How the JMS Adapter works
- Working with the JMS Adapter
- Tutorial 1: Using the JMS Adapter in Send-Only mode
- Tutorial 2: Using the JMS Adapter in Send/Receive mode
- Tutorial 3: Using the JMS Adapter in Send-Only and Receive-Only mode
- Tutorial 4: JMS Adapter - Receiving messages from external applications

JMS Adapter features

The JMS Adapter can operate in the following modes:
• **Send-Only mode.** In this mode, the adapter (shown in the figure below as the Send workstep) sends a message to the external system.

**Figure 9: Send-Only mode**

![Send-Only mode diagram](image)

After the message is sent, the workstep completes and process execution continues without delay.

• **Send/Receive mode.** In this mode, the adapter (shown in Figure 10: Send/Receive mode on page 100 as the SendReceive workstep) first sends a JMS message to the external system. After the message is sent, the workstep remains activated until a response is delivered.

**Figure 10: Send/Receive mode**

![Send/Receive mode diagram](image)

When the response is finally received, the adapter extracts the necessary information from the JMS message, the workstep completes and the process execution continues.

• **Send-Only and Receive-Only mode.** This mode is always used in combination of a Send-Only workstep and a subsequent Receive-Only workstep:

**Figure 11: Send-Only and Receive-Only workstep**

![Send-Only and Receive-Only workstep diagram](image)

The following procedures are typically implemented for Send-Only / Receive-Only modes.

• First, the adapter in the Send workstep composes and sends a message to the external system.
• After the message is sent, the workstep completes immediately and the process execution continues.

• At some point later in the process, the workflow reaches the workstep (in this case, Receive) that contains an adapter configured in Receive-only mode. If the response from the EIS has been received at the time when this workstep is activated, the adapter extracts the relevant information from the message and the workstep completes immediately. If the response is not yet available at the time the workstep is activated, the Receive workstep will remain in Active state until the message is received.

• **Receive-Only mode**. In this mode, the adapter only receives a message from the external system.

**Figure 12: Receive-Only mode**

After the message is received, the workstep completes and process execution continues without delay.

---

**How the JMS Adapter works**

The simplified diagram below shows the operation of the JMS Adapter along with the other components of the asynchronous adapter framework.

**Figure 13: JMS Adapter in an Asynchronous Adapter framework**
In this example, the JMS Adapter in the SendReceive workstep is shown performing a send/receive operation. First the adapter generates the outgoing message and sends it to the Outgoing queue. The queue name is specified when the adapter is configured.

The message is picked from the Outgoing queue up by the custom application which processes the message and delivers the response to the incoming queue. By default, the incoming messages are expected in a queue named "JMSAdapterQueue" but, if necessary, an additional Message Driven Bean (MDB) can be installed to accept messages delivered to any other queue.

The MDB, listening to the response queue, in turn passes the message to the Dispatcher, which is a component whose task is to deliver response messages to the installed asynchronous Managed Adapters. The Dispatcher delivers the response to the JMS Adapter for processing. After the necessary information is extracted from the message and is inserted into the process data slots, as defined in the JMS Adapter configuration and mapping stage, the JMS Adapter workstep is completed and process execution continues.

Working with the JMS Adapter

The JMS Adapter is a Managed Adapter and its usage is standardized along the same lines as the other Managed Adapters. For more information, see the adapter configuration and mapping framework described in Understanding the Managed Adapter framework on page 19

Assigning JMS Adapter as workstep performer

To assign the JMS Adapter as the performer of an Adapter workstep:

1. Click the Assign Participants link in the Tasks pane.
2. Expand Adapters > Managed > JMS Adapters.
3. Choose one of the following:
   • Drag the JMS Managed Adapter from the Tasks pane to the Content pane.
   • Right-click the Adapter workstep and select Change Performer from the pop-up menu. Select JMS Adapter from those listed in the Change Performer dialog box.

   **Note:** As with any other Managed Adapter, there may be several preconfigured instances for the JMS Adapter, each selected by choosing different adapter name.

4. Double-click the Adapter workstep to open the Properties view, where the JMSAdapter is now listed as the performer of the workstep.

Configuring the JMS Adapter

You can use a JMS Adapter in a BP Server application in any of the three modes namely, Send-Only, Send/Receive, and Receive-Only. However, you can use a JMS Adapter in a Web application only in Send-Only mode; it cannot be used in Send/Receive mode and Receive-Only modes.

The Configuration tab in the Adapter workstep Properties view includes the following options:

• Configure
• Changemapping
To perform the JMS Adapter configuration follow the steps:

1. Click **Configure**, to open the JMS Adapter Configurator.

   **Note:** You can also open the JMS Adapter Configurator using the new Managed Adapter Browser (available only from Progress Developer Studio for OpenEdge) functionality. In addition to configuring, you can use the Managed Adapter Browser to perform functions like creating a copy, renaming, deleting, as well as importing and exporting the configuration information. For more information, refer to the "Using the Managed Adapter Browser" section of the *Progress Developer Studio for OpenEdge User’s Guide*.

   The **JMS Adapter Configurator** has three tabs:

2. In the **Configuration** tab, specify general configuration parameters for the adapter, such as connection mode for the Adapter (Send, Send/Receive, and Receive options in the Adapter Mode list), connection factory, and destination (queue) name. Depending on the selected connection mode, the tabs Outgoing Message and Incoming Message may become disabled.

3. Click the **Outgoing Message** tab to define the format of the outgoing message. Currently, the JMS Adapter supports JMS Map message format, in which the message content is a set of name/value pairs. The Outgoing Message tab in turn includes two tabs:
   - **Payload**: Helps define the content of the message payload (body).
   - **Properties**: Lists message header properties that have to be set by the adapter.

4. Click the **Payload** tab to view a list of fields, each field associated with a parameter name and data type. To add a new parameter, click **Add**, to open the **Parameter Editor** dialog box. To modify existing parameters, select a parameter from those listed, click **Modify**, and make the required changes in the **Parameter Editor** dialog box.

   **Note:** The JMS Adapter supports several numerical data types that Business Process Server does not support, and these include: Integer, Byte, Float, and Short. There is the possibility of data loss for numerical data types if a Business Process Server data type is mapped to one of the JMS Adapter data types that is not supported in Business Process Server; and if the incoming data value for the Business Process Server data is more than the boundary value of the mapped JMS Adapter data type.

5. Click the **Properties** tab to view a list of properties that have to be explicitly set by the JMS Adapter in the outgoing message’s header.

   You can edit the header properties using the **Property Editor** dialog box.

6. Click the **Incoming Message** tab (enabled when the Adapter Mode is set to Send/Receive or Receive) to define the format of the incoming message (the response expected from the external system).

   Currently, the JMS Adapter requires the response message to be a JMS Map message, where the message body is a set of name/value pairs. Here, the list in the **Payload** tab defines a number of parameters that have to be extracted from the message body of the response. Similarly, the **Properties** tab includes the list of message properties that have to be extracted from the header.

   Note that while any property name can be manually entered, the name can be also selected from a list of predefined properties (the Name field is an edit-enabled combo box that provides a listing of various system properties). Selecting a predefined property also automatically adjusts the property type.

   **Note:** The list of property names is provided for your convenience only. Keep in mind that some message properties are read-only. The **Property Editor** dialog does not enforce correct property usage.
Defining mapping with the Mapping Configurator

When the JMS Adapter configuration, inputs, and outputs are defined, the mapping to actual dataslots is performed through the Mapping Configurator. The Mapping Configurator may look differently, depending on the way you configured the JMS Adapter. Generally, for a send/receive configuration, the Mapping Configurator has an Incoming Message tab, an Outgoing Message tab, and a Runtime Setup tab that allows you to map such additional parameters as Correlation ID and, optionally, other parameters that had been left empty during JMS Adapter configuration.

The Outgoing Message mapping tab includes two tabs used to map dataslots or constant values into the outgoing message body (payload) fields and header. Note that in order to generate the message successfully, you must map all of the fields in the Outgoing Message tab. Otherwise, the message cannot be constructed properly and you may get an error message while executing the JMS Adapter.

Similarly, the Incoming Message mapping tab also includes two tabs, where data from the response payload and header is mapped back to dataslots.

The last tab–Runtime Setup,–includes any additional run-time configuration that may be needed.

One such parameter is the Correlation ID. In certain situations (as described in the next section), the Correlation ID may need to be obtained from a dataslot, or the automatically generated ID may need to be stored in a dataslot for future use. You can define this mapping in the Runtime Setup tab.

Receiving messages: Passing the Correlation ID

When you use the JMS Adapter in Send-Only mode, the adapter’s task is simple: just generate the message using the data from process dataslots, and then send this message to the preconfigured JMSQueue. When a response is expected, there is an additional complication: out of the many incoming messages, you have to find the response to that particular message, and deliver it to the right adapter in the proper context (process instance and workstep). This is achieved using the message’s CorrelationID.

The CorrelationID is a unique string, identifying each message as part of a specific "conversation" between your business process and the external EIS. When a message is created and sent, the JMS Adapter generates this unique CorrelationID, which can be stored in a dataslot. Alternatively, you can define your own CorrelationID using static values or dataslots. Multiple dataslots can be used in the '@ds1:@ds2' format. If a response is expected in the same workstep, there is no need for you, as a process developer, to take further action—the adapter records the CorrelationID of the message it just sent and makes sure that the response with the same ID is delivered to it. However, if the response of a message is sent from one workstep, and you expect to receive the response in another workstep, the sending and receiving adapters need your help to coordinate their work.

When the message is sent by an adapter in the Send-Only workstep, the CorrelationID of the newly generated message is available to you as an optional adapter output. Make sure to map this output to a CHARACTER dataslot, so as to retain the CorrelationID.

Later, in the Receive-Only workstep, map this same dataslot to the CorrelationID input of the receiving adapter. This ensures that the adapter identifies the response message it is waiting for.

To thoroughly understand this configuration, take a look at the following tutorials:

• Tutorial 1: Using the JMS Adapter in Send-Only mode on page 105.
• Tutorial 2: Using the JMS Adapter in Send/Receive mode on page 108.
• Tutorial 3: Using the JMS Adapter in Send-Only and Receive-Only mode on page 111.
• Tutorial 4: JMS Adapter - Receiving messages from external applications on page 114.
Delivering messages to a receive-only workstep in a separate process

If you want to deliver a message to an independent business process that contains a Receive-Only workstep instead of passing the message from one workstep to another in the same process, perform the following procedures:

1. Make sure that the Correlation ID of the message and the Correlation ID of the receiving workstep is configured to expect a match. For an example, see Defining mapping to receive messages from external applications on page 116.

2. If you know the process instance ID and the name of the receiving workstep, you can set the JMS correlation ID of the message in the following format:

   \[ WS=\text{process\_instance\_id:workstep\_name} \]

   For example, a message with Correlation ID "WS=124:Adapter1" will be delivered to the workstep named Adapter1 in the process with Process Instance ID of "124".

3. In either of the preceding steps, make sure the JMS Adapter is properly configured in the Receive-Only mode.

Tutorial 1: Using the JMS Adapter in Send-Only mode

The goal of this tutorial is to provide detailed information on how to build applications that can generate and send JMS messages using the JMS Adapter.

The Send-Only mode is the simplest mode in which the JMS Adapter can be used. In this mode, the business process includes a single adapter workstep that generates a JMS message and posts it to a predefined JMS queue. No response of the message is expected.

Before proceeding, you should create a test JMS queue, accessible from your application server. We will use this queue to deliver all the test messages, generated from our application. Using the application server console, define a queue named Q1. The way you define a JMS queue will vary depending on your application server vendor and version. Refer to the application server documentation for more information.

The process we will develop will be used to publish a JMS message that contains some product information — product ID number, name and price. After each message is published, a confirmation workstep will be presented.

The following sections describe how to use the JMS Adapter to generate and send JMS messages in the Send-Only mode:

Defining the process workflow for the Send-Only mode

To define process workflow for send-only mode:

1. Create a new business process named "JMSAdapter_SendOnly."
2. Click the Dataslots content pane tab and define the following dataslots that are used to store our product information:

<table>
<thead>
<tr>
<th>Dataslot name</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProductID</td>
<td>INTEGER</td>
</tr>
</tbody>
</table>
3. Create a process workflow consisting of three worksteps as shown in the following figure.

**Figure 14: Process workflow for the Send-Only mode**

4. The ProductInfo workstep is used to enter the product information. Right-click the workstep, select **Properties**, and add the three dataslots we defined in Step 2.

5. Complete defining properties of all the human-performed worksteps, before configuring the JMS Adapter. Open the **Properties** view for the Confirmation workstep and add the Name dataslot in the same way as in Step 4.

6. Add and configure the "SendMessage" JMS adapter workstep, which actually generates and sends the message.

**Note:** The GenericJMSAdapter adapter instance is used to denote a non-preconfigured, generic JMS adapter. Sometimes, an adapter configuration may be quite complex, so you may wish to preserve it and reuse it across several processes. In this case, you can define your own adapter instances, that show in the Name field along with the GenericJMSAdapter.

### Configuring the JMS Adapter for Send-Only mode

To configure JMS adapter for send-only mode:

1. Click the **Configuration** tab of the JMS adapter workstep **Properties** view and then click **Configure** to open the **JMS Adapter Configurator** dialog box.

2. From the **Adapter Mode** drop-down list, select the **Send** option. By default, "jms/BMConnectionFactoryXA" is provided as the Connection Factory (JNDI Name); you can change this default name if required. Enter a target Queue Name (JNDI Name); in this case, enter Q1.

3. Since we have set the JMS Adapter to work in Send-Only mode, there are only outgoing messages, and responses are not expected. As a result, the **Incoming Message** tab is inactive.

4. Click the **Outgoing Message** tab to define the format of the message to be published.

5. Notice the two tabs, **Payload** and **Properties**. The **Payload** tab is used to define the parameters to be included in the message body (payload); and the **Properties** tab is used to define the format, or custom properties, in the message header. For this tutorial, let us configure all our product data to go in the message payload.

6. To start defining the message payload fields, click **Add**. The **Parameter Editor** dialog box displays.

7. Define a parameter named "ID" and select the data type as Long. Use this field to store the Product ID code when we publish our messages.
8. Similarly, create the rest of the fields. After defining all fields, the **Outgoing Message** format tab appears.

9. At this point, the basic JMS Adapter configuration is completed. Click **OK** to close the **JMS Adapter Configurator** and automatically open the **Dataslot Mapping** dialog box.

---

**Defining mapping for the Send-Only mode**

To define mapping for send-only mode:

1. After configuring the JMS Adapter, you can define the mapping for the Send-Only mode, indicating the correspondence between message fields and the actual dataslots used in your process. The **Dataslot Mapping** dialog box automatically opens after configuring, or you can click **Change mapping** in the **Configuration** tab to open this dialog box.

2. Select the source dataslots to match the JMS Adapter input parameters, as shown above. Make sure that no input parameter has been left empty—otherwise, when the adapter gets executed, it will not know how to generate the output message.

---

**Note:** The data types that can be used in the JMS Adapter configurator do not necessarily match the standard data types of the Business Process Server dataslots. Where there is a difference in the data types expected by the adapter (or returned from the adapter) and the mapped dataslot type, the run-time mapper will perform the necessary conversion. Refer to the "Supported Types for Conversion" table in the Developing Custom Managed Adapters chapter in the Customization Guide for more information about the data conversions supported by the mapper.

---

3. The **Runtime Setup** tab contains additional information, such as the message Correlation ID, that can be specified for the message. Although this tab is used for more complex configuration, it is not used in this case. Click **OK** to complete map configuration.

4. Save the SendMessage workstep properties and the process.

You can now publish and test the process. From Business Process Portal, monitor the process execution. If the application server allows it, check the content of the Q1 queue to verify that the message is sent.

---

**Troubleshooting for the Send-Only mode**

If you do not see the Confirmation task in your Task list, it means that there was a problem in the process—the most probable place is the SendMessage workstep. Check the following:

- Is the JMS Adapter configured correctly? Maybe you did not specify the mode correctly—selecting Send/Receive makes your adapter wait for a reply that never arrives. Check the JMS queue name—is the JNDI name correct?

- Is the queue accessible? Depending on the application server you use, make sure that the queue is accessible from the context where Business Process Server is running. For some application servers, you may need to restart the server before the queue becomes accessible.

- If the previous steps do not correct the problem, check the **OEBPS HOME\logs\managedadapter.log** file. Additional tracking information about the JMS Adapter operation may be available there.
Tutorial 2: Using the JMS Adapter in Send/Receive mode

Tutorial 1: Using the JMS Adapter in Send-Only mode on page 105 demonstrated how the JMS Adapter can be used to send messages, without considering what will happen with the messages later. When the message is sent, it is usually delivered to some kind of enterprise information system (EIS) that will process it and may respond with its own message.

In order to test the capabilities of the JMS Adapter to receive such responses, you can simulate the action of the EIS, preferably without involving any additional software. The easiest way to do this is to deliver the output message directly to the incoming JMS queue, thus bypassing the need for any external program.

We can use our previous example as a base and define a process that generates a message containing some product information and receives a response. Ideally, we should send this message to the EIS to process it and return a response, but since we are bypassing the EIS, the best we can do is to receive the message that we sent, and to treat it as the response we are waiting for.

Defining the process workflow for the Send/Receive mode

To define process workflow for send/receive mode:

1. Create a new business process named, "JMSAdapter_SendReceive."

2. In the Data slots content pane tab, define the following dataslots:

<table>
<thead>
<tr>
<th>Data slot name</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProductID</td>
<td>INTEGER</td>
</tr>
<tr>
<td>Name</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>Price</td>
<td>DECIMAL</td>
</tr>
<tr>
<td>R_ProductID</td>
<td>INTEGER</td>
</tr>
<tr>
<td>R_Name</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>R_Price</td>
<td>DECIMAL</td>
</tr>
</tbody>
</table>

The first three dataslots are used to store the same information as in Tutorial 3. The three new dataslots will be used to confirm the successful delivery of the message and that the information from the "response" is correctly received.

3. When you have defined all the dataslots, click the Diagram content pane tab and create the process workflow by adding the following worksteps.

Figure 15: Process workflow for the Send/Receive mode
4. The ProductInfo workstep is used to supply values for the ProductID, Name, and Price dataslots. Double-click the workstep to open the **Properties** view. Click the **Fields** tab and add these three dataslots.

5. The Confirmation workstep is used to verify that the response has been received and processed correctly by reading the R_ProductID, R_Name, and R_Price dataslots. The values of these dataslots are retrieved from the response message and should match the values that were given in the first workstep. Double-click the Confirmation workstep to open the **Properties** view. Open the **Fields** tab and add the R_ProductID, R_Name, and R_Price dataslots.

6. The SendReceive workstep is performed by the JMS Adapter, and is used to send the message and receive the response.

## Configuring the JMS Adapter for the Send/Receive mode

**To configure JMS adapter for send/receive mode:**

1. To configure the JMS Adapter, double-click the SendReceive workstep to open the **Properties** view. Click the \( \text{...} \) button in the **Performer** field and select the GenericJMSAdapter, as described in Step 6.

2. Click the **Configuration** tab of the SendReceive workstep **Properties** view and then click **Configure** to open the **JMS Adapter Configurator** dialog box.

3. Set the **Send/Receive** option from the Adapter Mode list, since we both send the message and receive the response in a single workstep. Keep the connection factory name "\text{jms/BMConnectionFactoryXA}". For the Queue Name (JNDI Name) enter "\text{jms/JMSAdapterQueue}"—this is the default "incoming" queue for the JMS Adapter. Note that you are trying to simulate the action of the EIS by directly sending the message to the incoming queue.

4. As we selected the **Send/Receive** option, both the **Outgoing Message** and **Incoming Message** tabs are enabled. Click the **Outgoing Message** tab and define the message format.

5. As there is no real client, the "response" that we receive is actually the same message we sent.

   Click the **Incoming Message** tab and define the message format as described for the outgoing message in Step 3. The completed incoming message appears.

6. Ensure that you use exactly the same parameter names and data types for both the outgoing and incoming messages.

7. Click **OK** to complete JMS Adapter configuration, and automatically open the **Dataslot Mapping** dialog box.

## Defining mapping for the Send/Receive mode

**To define mapping for send/receive mode:**

1. The **Dataslot Mapping** dialog box automatically opens after configuring. Alternatively, in the **Configuration** tab of the SendReceive workstep **Properties** view, you can click **Change mapping** to open the **Dataslot Mapping** dialog box (or the **Mapping Configurator**), where you can define the mapping between the adapter inputs/outputs and the process dataslots.

2. Select the ProductID, Name, and Price dataslots to supply values for the outgoing message. Make sure that all the input parameters are mapped to a dataslot, else the JMS Adapter is not able to generate the output message.

3. Click the **Incoming Message** tab to define the mapping for the response. Select the R_ProductID, R_Name, and R_Price dataslots to store the values from the message response.

4. Click **OK** to complete the dataslot mapping.
In our previous tutorial, when we used the JMS Adapter to only send a message, the Adapter workstep is "synchronous"—that is, the adapter is called by BP Server and, when the adapter execution (sending of the message) completes, BP Server automatically completes the Adapter workstep, so the process execution can continue.

**Defining the SendReceive workstep as asynchronous for the Send/Receive mode**

When a JMS Adapter is configured in Send/Receive mode, it is by default configured as asynchronous. In asynchronous mode, the send/receive workstep would not get completed immediately after the message is sent. Instead, the workstep would remain in active state until the response message is received.

**To define the workstep as asynchronous:**

1. From the **Advanced** tab in the SendReceive workstep **Properties** view, ensure that the **Wait for adapter to complete before processing** check box is not selected.
   
   This ensures that the workstep will be asynchronous and will remain activated until the response is received.

2. Save the process.

**Deploying and testing the JMS Adapter for the Send/Receive mode**

To deploy JMS adapter for send/receive mode:

1. Using the Admin Console, publish the process and start a process instance.

2. Monitor the process execution after sending the message.

3. Once you see the Confirmation task in your Task list, check if the values of the R_ProcessID, R_Name, and R_Price dataslots are same as the ProcessID, Name, and Price dataslots, respectively.

**Troubleshooting for the Send/Receive mode**

If something does not work as expected, check the following:

- Is the JMS Adapter installed properly? Check the `bpserver.log` file for errors. If the adapter is not installed properly, you may get a "ClassNotFoundException" error message.

- Is the JMS Adapter configured correctly? You may have not specified the mode correctly—selecting Send makes your adapter continue without waiting for a reply. Check the JMS queue name—is the JNDI name correct?

- Is the **Wait for adapter to complete before processing** check box set correctly? Failing to clear this checkbox causes your workstep to complete immediately after the message is sent and the R_ProcessID, R_Name, and R_Price dataslots are not updated from the response message.

- If the previous suggestions do not resolve your problem, check the `OEBPS_HOME\managedadapters\managedadapter.log` file. Additional tracking information about the adapter operation may be available there. Depending on the debug level you selected during the JMS Adapter installation, this file can contain a complete record for all sent and received messages. Use this record to figure out which step of the message delivery is failing.
Advanced exercises for the Send/Receive mode

As an advanced exercise for the Send/Receive mode, try writing a JMS client code. Writing a simple MDB that receives the messages, performs some data processing and responds with another JMS message that allows you to verify the message delivery, and to experiment with different format for the message response. When installing the MDB, make sure that it is receiving messages from your own queue—in no case use the system queue “JMSAdapterQueue”—this disrupts the normal work of all the JMS Adapters installed on the server, since your MDB competes with the JMS Adapter MDB for the responses. After selecting a queue, make sure that the same queue name is used in the JMS Adapter configuration.

Tutorial 3: Using the JMS Adapter in Send-Only and Receive-Only mode

This tutorial is used to demonstrate how a message, sent from one workstep, can be received at a later time by a different workstep.

In Tutorial 2: Using the JMS Adapter in Send/Receive mode on page 108, we have already configured the JMS Adapter to wait for a message response. The difference now is that the response is to be delivered to a different workstep, not the one that sent the message. This tutorial demonstrates how to ensure the correct message delivery by passing the correlation ID of the outgoing message to the receiving workstep, so that the correct response can be identified.

Defining the process workflow for the Send-Only and Receive-Only mode

To use the JMS Adapter in a Send-Only mode for one workstep, and then in a Receive-Only mode for a later workstep:

1. Create a new business process named "JMSAdapter_SendOnly_ReceiveOnly."
2. In the Dataslots content pane tab, define the following dataslots:

<table>
<thead>
<tr>
<th>Dataslot name</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProductID</td>
<td>INTEGER</td>
</tr>
<tr>
<td>Name</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>Price</td>
<td>DECIMAL</td>
</tr>
<tr>
<td>R_ProductID</td>
<td>INTEGER</td>
</tr>
<tr>
<td>R_Name</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>R_Price</td>
<td>DECIMAL</td>
</tr>
<tr>
<td>CorrelationID</td>
<td>CHARACTER</td>
</tr>
</tbody>
</table>
3. After defining all the dataslots, click the **Diagram** tab and create the process workflow as shown in the following figure.

**Figure 16: Process workflow with separate Send-Only and Receive-Only modes**

4. The ProductInfo workstep is used to supply values for the ProductID, Name, and Price dataslots. Double-click the workstep to open the **Properties** view. Click the **Fields** tab, and add these three dataslots.

5. The Send workstep is performed by the JMS Adapter, configured in Send-Only mode. Here, the outgoing message will be posted in JMS adapter queue, and the correlation ID for this message will be stored in a dataslot. Configuring the JMS Adapter is described in **Configuring the JMS Adapter for the Send workstep** on page 112.

6. The Confirmation1 workstep is used to confirm that the message has been sent successfully. Double-click the workstep to open the **Properties** view. Click the **Fields** tab, and add the CorrelationID dataslot to this workstep.

7. The Receive workstep is performed again by the JMS Adapter, this time configured in Receive-Only mode. Here, the JMS Adapter will receive the response, using the Correlation ID dataslot to identify the correct message. Configuring the JMS Adapter is described in **Configuring the JMS Adapter for the Receive workstep** on page 113.

8. Finally, the Confirmation2 workstep is used to verify that the response has been received and processed correctly by reading the R_ProductID, R_Name, and R_Price dataslots. The values of these dataslots are retrieved from the "response" message and should match the values that were given in the first workstep. Double-click the Confirmation2 workstep to open its **Properties** view. Click the **Fields** tab, and add the R_ProductID, R_Name, and R_Price dataslots.

9. After completing configuration of all human-performed workstep, you can now configure the JMS Adapter worksteps.

**Configuring the JMS Adapter for the Send workstep**

To configure JMS adapter for the Send workstep:

1. Double-click the Send workstep to open its **Properties** view.

2. Click the **Configuration** tab and then click **Configure**, to open the **Configurator** dialog box.
3. Select the Send option in the Adapter Mode list. Keep the "jms/BMConnectionFactoryXA" default for the Connection Factory (JNDI Name), and the "jms/JMSAdapterQueue" default for the Queue Name (JNDI Name).

4. Click the Outgoing Message tab and configure the message format.

5. Click OK to complete JMS Adapter configuration, automatically to open the Dataslot Mapping dialog box.

Mapping the send workstep dataslots

To map the Send workstep dataslots:

1. The Dataslot Mapping dialog box automatically opens after configuring. Alternatively, in the Configuration tab of the Send workstep Properties view, you can click Change mapping to define the mapping for the message payload.

2. If the JMS Adapter was just sending the message and no response was expected, this should be enough. However, in this case, the sending adapter must also store the correlation ID of the message in a dataslot, where it can be used by the receiving adapter. Click the Runtime Setup tab.

3. Select CorrelationID in the Target list so that the correlation ID of the message just sent will be stored in the CorrelationID dataslot.

4. Click OK to complete the mapping, then click OK to close the Properties dialog for the Send workstep.

Configuring the JMS Adapter for the Receive workstep

To configure the Receive workstep:

1. Double-click the Receive workstep to open its Properties view.

2. Verify that the JMS Managed Adapter is the performer. If not, click the ellipsis button beside the Performer field. Select the GenericJMS Adapter directly from the list of predefined adapters. Click OK.

3. Click the Configuration tab and then click Configure, to open the JMS Adapter Configurator dialog box.

4. Select the Receive option in the Adapter Mode list.

5. Click the Incoming Message tab. Configure the format of the response message in the same way as the outgoing message. Verify the field names and types to confirm that they match the message we are expecting.

6. Click OK to complete JMS Adapter configuration, and automatically open the Dataslot Mapping dialog box.

Mapping the Receive workstep dataslots

To map the Receive workstep dataslots:

1. The Dataslot Mapping dialog box automatically opens after configuring. Alternatively, in the Configuration tab of the Receive workstep Properties view, you can click Change mapping and define the mapping for the payload in the response message.

2. In this example, we are expecting the response to the message sent from a previous workstep. Here, we must supply the adapter with the proper correlation ID, so it can retrieve the correct response message. Open the Runtime Setup tab and select the CorrelationID dataslot in the Source field, as shown below.

3. Click OK to complete the map configuration.
Defining the Receive workstep as asynchronous for a Receive-Only mode

Since the response of the message may come at any time — before or after the Receive workstep has been activated—, we must make sure that this workstep is configured as asynchronous.

**To define a workstep as asynchronous for a receive-only mode:**

1. In the Advanced tab of the Properties view for the Receive workstep, make sure that the Wait for adapter to complete before processing check box is not checked.
2. Save the process under the name JMSAdapter_ReceiveOnly.

Deploying and Testing the JMS Adapter for Send-Only and Receive-Only modes

**To deploy JMS adapter for send-only and receive-only modes:**

1. Using the Admin Console, publish the process and start a process instance.
2. Monitor the process execution after sending the message. In the Confirmation1 workstep, verify that the CorrelationID was stored in the appropriate dataslot.
3. When the message is received, in the Confirmation2 workstep, you should see the correct values for the R_ProductID, R_Name, and R_Price dataslots.

Troubleshooting for Send-Only and Receive-Only modes

If you encounter problems in this tutorial, check the following:

- Is the CorrelationID passed correctly? Revisit the mapping for the Send and Receive worksteps; check the value in the CorrelationID dataslot after the Send workstep completed.

- Did you define the message formats correctly? The format of the outgoing and incoming messages should match. Check the adapter configuration for both Send and Receive worksteps. If you find and correct any errors in the adapter configuration, make sure to run the map configuration again for each workstep, otherwise your fixes may not be active.

- Confirm that the Wait for adapter to complete before processing checkbox setting is correct. The box must be checked for the Send workstep, and cleared for the Receive workstep.

Tutorial 4: JMS Adapter - Receiving messages from external applications

This tutorial demonstrates how an external application can send messages to one of your business processes, without knowing about the process, receiving workstep, and other details.
In the following example, an Order Processing Application receives orders, and generates and sends JMS messages containing order data. The messages are received by a process containing the JMS Adapter.

**Figure 17: Sample workflow for receiving messages from external applications**

The Order Processing Application does not have knowledge about your process; it does not have to know the Process Instance name, nor the workstep receiving the message. It is expected, however, to generate the JMS message in a format supported by the JMS Adapter, and to use the OrderID as JMS message correlation ID. When the process is started, it is provided an OrderID (either manually by the user starting to process an order, or through an adapter) and Business Process Server will deliver the message to the JMS Adapter workstep.

### The Order Processing application

The design of your Order Processing Application depends on: your operating environment, the type of application server and JMS server, how the application will receive the order data, and other factors. In this tutorial, we assume that your application produces MapMessages with a single string field, named OrderData, and the CorrelationID of each message is set to match the Order ID:

```java
MapMessage msg = jmssession.createMapMessage();
msg.setString("OrderData", data);
msg.setJMSCorrelationID(correlationId);
```

If you plan to actually implement the application (rather than just reading the example below), refer to the documentation of your application server and the many examples of JMS sender applications available on the Internet.

### Defining the process workflow for receiving messages from external applications

**To define process workflow for receiving messages from external applications:**

1. Create a new application with the following dataslots:

<table>
<thead>
<tr>
<th>Dataslot name</th>
<th>Data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>OrderID</td>
<td>CHARACTER</td>
</tr>
<tr>
<td>OrderData</td>
<td>CHARACTER</td>
</tr>
</tbody>
</table>

For simplicity, we assume all the data about an order is stored in the OrderData dataslot.

2. After defining the dataslots, click the **Diagram** tab and create the process flow, as shown in the figure below.

**Figure 18: Process template for receiving messages from external applications**
The Start workstep is used to manually enter the ID of the order that is being processed. The Receive workstep is performed by the JMS Adapter to retrieve the message from JMS and extract the order data. Finally, the order data is presented to the user in the ShowOrder workstep.

3. To configure the human-performed worksteps:
   a) Double-click the Start workstep to open the Properties view. Click the Fields tab and add the OrderID dataslot as workstep output. Click OK.
   b) Similarly, include both OrderID and OrderData dataslots in the ShowOrder workstep.

Configuring the JMS Adapter to receive messages from external applications

To configure JMS adapter to receive messages from external applications:

1. Double-click the Receive workstep to open its Properties view.
2. Click the Configuration tab of the Properties view and then click Configure to open the JMS Adapter Configurator dialog box. Set the Adapter Mode to Receive.
3. Click the Incoming Message tab and configure the message format.
4. The message includes only one string field, named data, that contains the order information.
5. Click OK to complete adapter configuration, and automatically open the Dataslot Mapping dialog box.

Defining mapping to receive messages from external applications

To define mapping to receive messages from external applications:

1. The Dataslot Mapping dialog box automatically opens after configuring. Alternatively, in the Configuration tab of the Receive workstep Properties view, you can click Change mapping to open the Dataslot Mapping dialog box, where you can define the mapping for the message payload.
2. Map the data parameter from the message to the OrderData dataslot.
3. Since we configured the adapter in Receive-Only mode, we must supply the CorrelationID of the expected JMS message. Click the Runtime Setup tab, and configure the OrderID dataslot as the source of the CorrelationID parameter.
4. Click OK to complete the mapping.
5. Since the response of the message may come at any time — before or after the Receive workstep has been activated —, we must make sure that this workstep is configured as asynchronous. In the Advanced tab of the Properties view for the Receive workstep, make sure that the Wait for adapter to complete before processing box is not checked.
6. Save the process under the name OrderProcessing.

Troubleshooting for receiving messages

If you encounter problems in this tutorial, check the following:

- Is the Correlation ID (Order ID) passed correctly? Revisit the mapping for the Start and Receive worksteps; make sure the Correlation ID is set properly in the JMS message.
• Confirm that the **Wait for adapter to complete before processing** checkbox is cleared for the Receive workstep.
OpenEdge Managed Adapter

The OpenEdge Adapter enables you to integrate Business Process Server applications with OpenEdge solutions. You can use OpenEdge Adapter as a participant to perform tasks in a workstep.

The OpenEdge Adapter is one of the Managed Adapters conforming to the Adapter Configuration and Mapping Framework. For more information regarding the managed adapter framework, see Understanding the Managed Adapter framework on page 19.

The OpenEdge managed adapter workstep in your process enables you to request OpenEdge business logic to be executed on the OpenEdge application server. This allows you to link a specific business process step to the execution of specific OpenEdge business logic.

For details, see the following topics:

• OpenEdge Adapter features
• Working with the OpenEdge Adapter

OpenEdge Adapter features

The OpenEdge Adapter enables you to:

• Use OpenEdge procedures with other processes in a BPM Workflow.
• Interact with OpenEdge procedures using BIZOE (Progress Interface Definition Language) files.
• Assign procedure file parameters to dataslots.

You can customize the OpenEdge adapter workstep to:
• Connect and log on to the OpenEdge application server.
• Run an external procedure in a state-free application server.
• Execute a single-run for an external procedure in a state-managed or state-free application server.
• Define a dataslot with ABL-specific data type to support "null" value.
• Map dataslots (ABL-specific and native) with parameters of type, input, output, and input/output.
• Support a return value for an external (or internal) procedure and a user-defined function.

Working with the OpenEdge Adapter

The OpenEdge adapter facilitates the use of OpenEdge procedures in your process workflow. You can use a OpenEdge procedure file or a generated BIZOE file for that procedure, in the added OpenEdge adapter workstep in your process.

**Note:** If you have installed Progress Developer Studio for OpenEdge, you can create a OpenEdge procedure file and generate the BIZOE file. For information regarding these operations, see the *Progress Developer Studio for OpenEdge Guide*.

You can configure and map the OpenEdge adapter to identify the OpenEdge application server, including the connection information, the procedure call, and mapping of the assigned dataslots with the parameters of the procedure.

**Note:** You can view information of the OpenEdge adapter including its configuration details, application server connection settings, and runtime properties in the `spring.xml` file (available after building the project). You can view this file in the Project Explorer view in the `<Process_name>\maps\<OpenEdge_adapter_workstep_name>` folder.

**To execute the OpenEdge adapter, you need to perform the following operations:**

1. Add and configure the OpenEdge adapter workstep, as discussed in Configuring the OpenEdge Adapter on page 120.
2. Define the dataslot mapping, as discussed in Defining the dataslot mapping on page 127.

Configuring the OpenEdge Adapter

You can add and configure the OpenEdge adapter workstep in your process by performing the following procedures.

**To configure the OpenEdge adapter:**

1. Click the Assign Participants link in the Tasks pane. Expand the Adapters > Managed > OEAdapters folder and drag the default GenericOEAdapter (or a previously configured OpenEdge adapter instance) to your Content pane, creating a OpenEdge adapter workstep.

2. Double-click the workstep to open its Properties view, and click the Configuration tab. This tab includes the Configure button and the Change mapping button for defining the adapter configuration and mapping respectively.

3. Click Configure to open the configurator box of the OpenEdge Adapter Configurator.
Note: You can also open the OpenEdge Adapter Configurator using the Managed Adapter Browser (available only from Progress Developer Studio for OpenEdge) functionality. In addition to configuring, you can use the Managed Adapter Browser to perform functions like creating a copy, renaming, deleting, as well as importing and exporting the configuration information. For more information, refer to the "Using the Managed Adapter Browser" section of the OpenEdge Getting Started: Developing BPM Applications with Developer Studio.

4. You can use the Input file box to load a BPM invocation (BIZOE) or an R-code (*.r) file.

Note: You can also load an OpenEdge procedure (*.p) file if you have installed Progress Developer Studio for OpenEdge.

- To load a file from a file directory (local or remote machine), click File System and select the file to be loaded as the input file.
- To load a file from the project workspace, click Workspace and select the file available in the project workspace. This functionality is available only if you have installed Progress Developer Studio for OpenEdge with BPM modules.
- After loading a file, if you modify the specified file, click Reload to reload the updated file.

5. You can use the Service Connection Details section to set (or modify) the connection details (as described in the following table) to the OpenEdge application server.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host name</td>
<td>To specify the host machine name (local or remote) of the OpenEdge application server.</td>
</tr>
<tr>
<td>Service name/Port</td>
<td>To specify the port address or service name of the host machine.</td>
</tr>
<tr>
<td>Application service</td>
<td>Modify (if required) the name of the application service.</td>
</tr>
<tr>
<td>User name/ Password</td>
<td>To enter the user name and password to connect to the specified OpenEdge application server. These fields are enabled only if you select the Session-managed option in the Session mode drop-down list.</td>
</tr>
<tr>
<td>AppServer Info</td>
<td>This box is enabled only if you select the Session-managed option in the Session mode drop-down list. Specify a user-defined character string to be passed to the AppServer connect procedure, which is executed each time a client connects to the AppServer.</td>
</tr>
<tr>
<td>Protocol</td>
<td>Select the protocol from this drop-down list. Available options are http and AppServer.</td>
</tr>
</tbody>
</table>
Available options are **Session-free** and **Session-managed**.

*Note:* For information regarding operating models and session model types in OpenEdge application server, see *OpenEdge Application Server: Developing AppServer applications.*

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session model</td>
<td>Available options are Session-free and</td>
</tr>
<tr>
<td></td>
<td>Session-managed.</td>
</tr>
<tr>
<td></td>
<td><em>Note:</em> For information regarding operating</td>
</tr>
<tr>
<td></td>
<td>models and session model types in OpenEdge</td>
</tr>
<tr>
<td></td>
<td>application server, see *OpenEdge Application</td>
</tr>
<tr>
<td></td>
<td>Server: Developing AppServer applications.*</td>
</tr>
<tr>
<td>Use SSL</td>
<td>Select this checkbox if you want to use SSL</td>
</tr>
<tr>
<td></td>
<td>authentication.</td>
</tr>
<tr>
<td>Direct connect</td>
<td>Select this checkbox if the -H and -S parameters</td>
</tr>
<tr>
<td></td>
<td>are interpreted as the network address and TCP/IP</td>
</tr>
<tr>
<td></td>
<td>port number of the AppServer connection. <em>Note:</em></td>
</tr>
<tr>
<td></td>
<td>This checkbox is disabled if you selected http</td>
</tr>
<tr>
<td></td>
<td>from the Protocol menu.</td>
</tr>
</tbody>
</table>

6. Click **Runtime Properties** to open the **Runtime Properties** dialog box, which display the runtime properties with their default values. The available properties vary depending on the session model selected in the **Session model** drop-down list. Modify the default values, if required. A valid certificate should be present in <dlc>\certs\psccerts.jar. If a valid certificate is not present, in the adapter **Runtime Properties** change **No host verification** property value to 1. For more information regarding the runtime properties, see *OpenEdge runtime properties* on page 122.

7. In the **Procedure Details** section, you can view the contents of the selected input file. The **Show members without annotations** and the **Generate annotations** checkboxes are enabled only if you have selected a procedure (*.p) file as the input file.

   - Select the **Show members without annotations** checkbox to list all the procedures and functions without annotations in the right pane of the **Procedure Details** section. If the main procedure file contains internal procedures and functions, the left pane of the **Procedure Details** section lists these elements in a tree structure.
   
   - Select the **Generate annotations** checkbox to generate annotations for a procedure or function in the right pane of the **Procedure Details** section. The procedure or function files are automatically updated with the annotations. If you generate annotations for a procedure or a function that is open in the OpenEdge Editor, the annotations are reflected when you save the open procedure or function file. If you select the **Show members without annotations** checkbox, the **Generate annotations** checkbox is also selected.

8. Click **OK** to save the save your changes in **OpenEdge Adapter Configurator** dialog box. The **Dataslot Mapping** dialog box appears.

**OpenEdge runtime properties**

*Table 16: OpenEdge Runtime properties* on page 123 details the OpenEdge runtime properties including its name, label, default value, and description.
Table 16: OpenEdge Runtime properties

<table>
<thead>
<tr>
<th>Property label</th>
<th>Property name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stale object</td>
<td>stale04GLOObject</td>
<td>0</td>
<td>The maximum duration (in seconds) that a service object (AppObject, SubAppObject, or ProcObject) can be idle before it is released.</td>
</tr>
<tr>
<td>timeout</td>
<td>Timeout</td>
<td></td>
<td>As part of managing certain service objects with OpenEdge, clients explicitly create them using factory methods before invoking other methods on them. When the client no longer requires the object, it has the responsibility to release the object from the service runtime context. However, if this time-out expires before the client releases the object, the OpenEdge Adapter assumes that the client application no longer requires access to the object, and deletes it from the service run-time context automatically. In effect, the adapter uses this time-out to provide garbage collection on service objects that client applications stop referencing and fail to release in the specified period of time. Any subsequent attempt by a client to access this object returns an error from the adapter.</td>
</tr>
<tr>
<td>Request wait</td>
<td>requestWait</td>
<td>-1</td>
<td>Determines how the adapter handles requests when the service connection pool is full. The connection pool is full when the number of active sessions is equal to the value of the maxSessions property and all sessions are currently running requests.</td>
</tr>
<tr>
<td>timeout</td>
<td>Timeout</td>
<td></td>
<td>• For value equal to -1, the OpenEdge adapter queues the request indefinitely till an AppServer session is available.</td>
</tr>
<tr>
<td>NameServer</td>
<td>nsClientMinPort</td>
<td>0</td>
<td>The minimum value for the adapter to specify for the UDP port number used to communicate with the NameServer. The value must be less than or equal to the value of the nsClientMaxPort property. For value equal to 0, the OpenEdge adapter chooses the NameServer client port number randomly. This property applies only to services that use a NameServer to access application services (an AppServer).</td>
</tr>
<tr>
<td>client min port</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Property label</td>
<td>Property name</td>
<td>Default value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------------------</td>
<td>---------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NameServer client max port</td>
<td>nsClientMaxPort</td>
<td>0</td>
<td>The maximum value for the adapter to specify for the UDP port number used to communicate with the NameServer. The value must be less than or equal to the value of the nsClientMaxPort property. For value equal to 0, the OpenEdge adapter chooses the NameServer client port number randomly. This property applies only to services that use a NameServer to access application services (an AppServer).</td>
</tr>
<tr>
<td>NameServer client port retry</td>
<td>nsClientPortRetry</td>
<td>3</td>
<td>The maximum number of requests that the adapter makes for a valid local UDP port number when attempting to communicate with the NameServer. This property applies only to services that use a NameServer to access application services (an AppServer).</td>
</tr>
<tr>
<td>NameServer client port retry interval</td>
<td>nsClientPortRetryInterval</td>
<td>200</td>
<td>The interval (in milliseconds) that the adapters waits between requests to get a valid UDP port number when attempting to communicate with the NameServer. This property applies only to services that use a NameServer to access application services (an AppServer).</td>
</tr>
<tr>
<td>Service logging level</td>
<td>serviceLoggingLevel</td>
<td>2</td>
<td>The amount and type of information written by the adapter to the service log for each log entry. Logging levels range from 1 to 4.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• For logging level equal to 1, the OpenEdge adapter logs the errors internally detected by the adapter only.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• For logging level equal to 2, the OpenEdge adapter logs additional errors that result in SOAP faults returned to the client.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• For logging level equal to 3, the OpenEdge adapter logs additional debugging information useful to the publisher.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• For logging level equal to 4, the OpenEdge adapter logs additional debugging information useful for Progress support services and engineering.</td>
</tr>
<tr>
<td>Service logging entry types</td>
<td>serviceLoggingEntryTypes</td>
<td>-</td>
<td>Specifies the type of information recorded when performing diagnostic logging. The value is a comma-separated list of valid entry types. It is also possible to specify a different logging level for each entry type. For example: &quot;ENTRY:2&quot;, where the number following the colon is the level. Levels for types must be greater than 1.</td>
</tr>
<tr>
<td>Property label</td>
<td>Property name</td>
<td>Default value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------</td>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Service fault level</td>
<td>serviceFaultLevel</td>
<td>2</td>
<td>The amount (level) of information returned to the client for a SOAP Fault as determined by an integer value. A level of 2 returns basic information in the <code>&lt;FaultCode&gt;</code> and <code>&lt;FaultString&gt;</code> elements for each SOAP Fault message, which is suitable for normal production environments. A level of 3 returns more detailed information that is suitable for development environments. Other values provide varying levels of diagnostic information, and are reserved for use by Progress Technical Support and Engineering.</td>
</tr>
<tr>
<td>Wait if service is busy</td>
<td>waitIfBusy</td>
<td>0</td>
<td>An integer value that determines how to handle client requests to a service that is busy processing a prior request. If the value is 1, the OpenEdge Adapter queues multiple requests for this service and executes them one at a time until the queue is empty. If the value is 0 and the adapter is executing a prior request for the service, each subsequent request for the same service fails until the adapter completes the request it is currently executing.</td>
</tr>
<tr>
<td>Connection lifetime</td>
<td>connectionLifetime</td>
<td>0</td>
<td>The maximum lifetime (in seconds) of AppServer connections in the connection pool for this service. If the value is 0, then the lifetime of these connections is unlimited, unless they are disconnected according to the requirements of other property settings, such as idleSessionTimeout. If the value is greater than 0, then the OpenEdge adapter maintains any AppServer connections for this service for the specified number of seconds. Thus, when the idleSessionTimeout interval expires, the adapter trims the connections in the connection pool beginning with those whose connectionLifetime interval has expired. However, the adapter maintains the connections for all services whose connectionLifetime interval has not yet expired regardless of other property settings.</td>
</tr>
<tr>
<td>No host verification</td>
<td>noHostVerify</td>
<td>0</td>
<td>If the value is set to 1, turns off host verification for an SSL Web service connection (specified by the appServiceProtocol property). If cleared, the adapter compares the host name of the connecting AppServer with the Common Name specified in the server digital certificate, and raises a Web service error if they do not match. With this parameter specified, the Web service never raises the error.</td>
</tr>
<tr>
<td>No session reuse</td>
<td>noSessionReuse</td>
<td>0</td>
<td>If set to 1, the Web service connection does not reuse the SSL session ID when reconnecting to the same AppServer for an SSL Web service connection (specified by the appServiceProtocol property).</td>
</tr>
</tbody>
</table>
### Description

Indicates if the client would employ the AppServer Keepalive protocol on this connection, if supported and enabled by the AppServer. To enable the protocol, specify the value as `allowServerASK`. To disable the protocol, specify the value as `denyServerASK`. The absence of this property indicates that the default value for the ServerASK protocol is used on this connection.

### Client ASK activity timeout

Determines the time interval the AppServer takes active steps to determine if the client is still connected. The ClientASK protocol denotes that the direction of the keepalive messages is from the client to the servers.

### Client ASK response timeout

Determines the time interval the AppServer can receive messages from the server. The ClientASK protocol denotes that the direction of the keepalive messages is from the client to the servers.

### SSL Ciphers

This parameter determines the encryption to be used to connect to the AppServer. The default value is "Default" and it stands for TLSv1.

### SSL Protocol

This parameter determines the security protocol to be used to connect to the AppServer. The default value is "Default" and it stands for TLSv1.

### Properties available only for "Session-free" session model

The minimum number of connected sessions allowed in the connection pool. The OpenEdge adapter attempts to keep at least this many sessions connected to the application service (AppServer).

The maximum number of connected sessions allowed in the service connection pool. Once the number of sessions in the pool reaches this limit, the OpenEdge adapter creates no additional sessions for this service, and handles all requests for this service according to the `requestWaitTimeout` property setting. A value of 0 indicates that the size of the connection pool is unlimited.

The number of network sessions to be created (and shared by all clients) when the service connection pool is initialized by the adapter. This value must be between the value of the `minSessions` property and the `maxSessions` property, inclusive, unless `maxSessions` is set to 0. If `maxSessions` is set to 0, the `initialSessions` value must only be greater than or equal to `minSessions`.

<table>
<thead>
<tr>
<th>Property label</th>
<th>Property name</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppServer keep alive</td>
<td>appServer KeepAlive</td>
<td>denyClientASK, allowServerASK</td>
<td>Indicates if the client would employ the AppServer Keepalive protocol on this connection, if supported and enabled by the AppServer. To enable the protocol, specify the value as <code>allowServerASK</code>. To disable the protocol, specify the value as <code>denyServerASK</code>. The absence of this property indicates that the default value for the ServerASK protocol is used on this connection.</td>
</tr>
<tr>
<td>Client ASK activity timeout</td>
<td>clientASKActivity Timeout</td>
<td>60</td>
<td>Determines the time interval the AppServer takes active steps to determine if the client is still connected. The ClientASK protocol denotes that the direction of the keepalive messages is from the client to the servers.</td>
</tr>
<tr>
<td>Client ASK response timeout</td>
<td>clientASKResponse Timeout</td>
<td>60</td>
<td>Determines the time interval the AppServer can receive messages from the server. The ClientASK protocol denotes that the direction of the keepalive messages is from the client to the servers.</td>
</tr>
<tr>
<td>SSL Ciphers</td>
<td>sslcipher</td>
<td>Default</td>
<td>This parameter determines the encryption to be used to connect to the AppServer. The default value is &quot;Default&quot; and it stands for TLSv1.</td>
</tr>
<tr>
<td>SSL Protocol</td>
<td>sslprotocol</td>
<td>Default</td>
<td>This parameter determines the security protocol to be used to connect to the AppServer. The default value is &quot;Default&quot; and it stands for TLSv1.</td>
</tr>
<tr>
<td>Minimum number of sessions</td>
<td>minSessions</td>
<td>1</td>
<td>The minimum number of connected sessions allowed in the connection pool. The OpenEdge adapter attempts to keep at least this many sessions connected to the application service (AppServer).</td>
</tr>
<tr>
<td>Maximum number of sessions</td>
<td>maxSessions</td>
<td>0</td>
<td>The maximum number of connected sessions allowed in the service connection pool. Once the number of sessions in the pool reaches this limit, the OpenEdge adapter creates no additional sessions for this service, and handles all requests for this service according to the <code>requestWaitTimeout</code> property setting. A value of 0 indicates that the size of the connection pool is unlimited.</td>
</tr>
<tr>
<td>Initial number of sessions</td>
<td>initialSessions</td>
<td>1</td>
<td>The number of network sessions to be created (and shared by all clients) when the service connection pool is initialized by the adapter. This value must be between the value of the <code>minSessions</code> property and the <code>maxSessions</code> property, inclusive, unless <code>maxSessions</code> is set to 0. If <code>maxSessions</code> is set to 0, the <code>initialSessions</code> value must only be greater than or equal to <code>minSessions</code>.</td>
</tr>
</tbody>
</table>
### Property label | Property name | Default value | Description
--- | --- | --- | ---
Idle session timeout | idleSessionTimeout | 0 | The duration (in seconds) between attempts by the adapter to shut down extra network connections to the AppServer, based on the client demand. The adapter monitors the maximum number of sessions needed since the last time-out, then disconnects any connections in excess of that number. A value of 0 indicates that the adapter will never disconnect idle sessions unless the connectionLifetime interval has expired.

NameServer client pick list size | nsClientPicklistSize | 8 | The number of available AppServer options (the broker pick list) that the adapter requests from the NameServer each time it looks up a given application service name. This property applies only to services that use a NameServer to access application services (an AppServer).

NameServer client pick list expiration | nsClientPicklistExpiration | 300 | The maximum duration (in seconds) that the adapter retains a list of AppServer options (pick list) for an idle application service. A value of 0 indicates that the pick list never expires. This property applies only to services that use a NameServer to access application services (an AppServer).

Minimum idle connections | minIdleConnections | 0 | If set to 1, turns off host verification for an SSL Web service connection (specified by the appServiceProtocol property). If cleared, the adapter compares the host name of the connecting AppServer with the Common Name specified in the server digital certificate, and raises a Web service error if they do not match. With this parameter specified, the Web service never raises the error.

### Defining the dataslot mapping

After configuring the OpenEdge Adapter workstep, you must define the mapping of your process dataslots (source and target) to the parameters in the selected OpenEdge file. The Dataslot Mapping dialog box displays the parameters depending on the input file selected in the OpenEdge Adapter Configurator dialog box.

**To define the dataslot mapping for an OpenEdge adapter:**

1. After adding and configuring the OpenEdge adapter workstep, double-click the workstep to open its Properties view.
2. From the Configuration tab, click Change mapping to open the Dataslot Mapping dialog box.
3. In the Source drop-down list in the Type Mapping tab, select the dataslot you want to associate with the parameter listed in the Parameter column. The value of this parameter is set from the value of the dataslot selected as the source.

**Note:** For information regarding the supported dataslot types (native or ABL) that can be mapped to each ABL parameter type, see Supported ABL dataslots for OpenEdge parameters on page 128.
4. In the **Target** drop-down list in the **Type Mapping** tab, select the dataslot you want to associate with the parameter listed in the **Parameter** column. The value of this parameter is stored in the dataslot selected as the target.

5. The **Advanced** tab is available only if you select the **Session-managed** option for session model in the **OpenEdge Adapter Configurator** dialog box. You can use this tab to specify dataslot values for AppServer parameters, namely, Username, Password, AppServer Info, and Return Value, which are used to connect to the AppServer.
   
a) In the **Source** drop-down list in the **Advanced** tab, select the dataslots you want to associate with the Username, Password, and AppServer Info parameters.

b) In the **Target** drop-down list in the **Advanced** tab, select the dataslot you want to associate with the Return Value parameter.

   At runtime, the value of the dataslot that you specify in **Advanced** tab replaces the character string value that you specify for the corresponding field in the OpenEdge adapter configuration dialog box.

6. Click **OK** to complete the dataslot mapping for the OpenEdge adapter workstep.

**Supported ABL dataslots for OpenEdge parameters**

You can use the **Dataslot Mapping** dialog box to map the ABL dataslot type as source, target, or both to the OpenEdge parameter types.

Table 17: Conversion Matrix of source dataslot type to OpenEdge parameter type on page 129 details the data conversion matrix between source dataslot types (native or ABL) and its supported OpenEdge parameter types used in ABL procedures.
Table 17: Conversion Matrix of source dataslot type to OpenEdge parameter type

<table>
<thead>
<tr>
<th>OpenEdge parameter type</th>
<th>Supported dataslot types</th>
<th>Level of support</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHARACTER</td>
<td>CHARACTER</td>
<td>Fully supported. Supported for a maximum of 30000 characters.²</td>
</tr>
<tr>
<td></td>
<td>INTEGER</td>
<td>Supported with string value of the dataslot passed to CHARACTER.</td>
</tr>
<tr>
<td></td>
<td>INT64</td>
<td>Supported with string value of the dataslot passed to CHARACTER.</td>
</tr>
<tr>
<td></td>
<td>DECIMAL</td>
<td>Supported with string value of the dataslot passed to CHARACTER.</td>
</tr>
<tr>
<td></td>
<td>DATETIMETZ</td>
<td>Supported with string value of the dataslot passed to CHARACTER.</td>
</tr>
<tr>
<td></td>
<td>LOGICAL</td>
<td>Supported with string value of the dataslot passed to CHARACTER.</td>
</tr>
<tr>
<td>CHARACTER LIST</td>
<td>LIST</td>
<td>Fully supported.</td>
</tr>
<tr>
<td>LONGCHAR</td>
<td>CHARACTER</td>
<td>Fully supported.²</td>
</tr>
<tr>
<td>DATE</td>
<td>DATETIMETZ</td>
<td>Supported for passing the date part.²</td>
</tr>
<tr>
<td>DATETIME</td>
<td>DATETIMETZ</td>
<td>Supported for passing the date and time part.²</td>
</tr>
</tbody>
</table>

² The UNKNOWN value is not supported for some dataslot types.
<table>
<thead>
<tr>
<th>OpenEdge parameter type</th>
<th>Supported dataslot types</th>
<th>Level of support</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATETIME-TZ</td>
<td>DATETIMETZ</td>
<td>Fully supported.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Supports the following formats to map a constant value with a DATETIME-TZ parameter in the OpenEdge Adapter Dataslot Mapping dialog:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• &quot;yyyy-MM-dd'T'HH:mm:ss.SSS(+-)zz:zz&quot; or &quot;yyyy-MM-dd HH:mm:ss.SSS(+-)zz:zz&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• &quot;yyyy-MM-dd'T'HH:mm:ss.SSS&quot; or &quot;yyyy-MM-dd HH:mm:ss.SSS&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• &quot;yyyy-MM-dd'T'HH:mm:ss&quot; or &quot;yyyy-MM-dd HH:mm:ss&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• &quot;yyyy-MM-dd&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For example, 1995-01-17T12:12:12.234+05:30 or 1995-01-17 12:12:12.234+05:30.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> The list of supported date formats specified above is applicable to these three date parameters: DATE, DATETIME, and DATETIME-TZ.</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>DECIMAL</td>
<td>Supported for numbers with a maximum of 50 digits with up to 10 digits after the decimal point.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DECIMAL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fully supported.</td>
</tr>
<tr>
<td></td>
<td>INT64</td>
<td>Fully supported.</td>
</tr>
<tr>
<td></td>
<td>INTEGER</td>
<td>Fully supported.</td>
</tr>
<tr>
<td>HANDLE</td>
<td>Handle</td>
<td>Fully supported.</td>
</tr>
<tr>
<td>INT64</td>
<td>INT64</td>
<td>Fully supported.</td>
</tr>
<tr>
<td></td>
<td>INTEGER</td>
<td>Fully supported.</td>
</tr>
<tr>
<td>INTEGER</td>
<td>INTEGER</td>
<td>Fully supported.</td>
</tr>
<tr>
<td>LOGICAL</td>
<td>LOGICAL</td>
<td>Fully supported.</td>
</tr>
<tr>
<td>MEMPTR</td>
<td>Memptr</td>
<td>Fully supported.</td>
</tr>
<tr>
<td>RAW</td>
<td>Raw</td>
<td>Fully supported.</td>
</tr>
<tr>
<td>ROWID</td>
<td>Rowid</td>
<td>Fully supported.</td>
</tr>
</tbody>
</table>
Table 18: Conversion Matrix of OpenEdge parameter type to target dataslot types on page 132 details the data conversion matrix between OpenEdge parameter types used in ABL procedures and the supported dataslot types (native or ABL) specified as target.
<table>
<thead>
<tr>
<th>OpenEdge parameter type</th>
<th>Supported dataslot types</th>
<th>Level of support</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHARACTER</td>
<td>CHARACTER</td>
<td>Fully supported. Supported for a maximum of 30000 characters.</td>
</tr>
<tr>
<td></td>
<td>INTEGER</td>
<td>Supported with conversion into numeric format.</td>
</tr>
<tr>
<td></td>
<td>INT64</td>
<td>Supported with conversion into numeric format.</td>
</tr>
<tr>
<td></td>
<td>DECIMAL</td>
<td>Supported with conversion into numeric format.</td>
</tr>
<tr>
<td></td>
<td>DATETIMETZ</td>
<td>Supported with conversion into numeric format.</td>
</tr>
<tr>
<td></td>
<td>LOGICAL</td>
<td>Supported with conversion into numeric format.</td>
</tr>
<tr>
<td>CHARACTER LIST</td>
<td>LIST</td>
<td>Fully supported.</td>
</tr>
<tr>
<td>LONGCHAR</td>
<td>CHARACTER</td>
<td>Supported for a maximum of 256 characters for a STRING of CHAR type and 2000 characters for a STRING of VARCHAR type.</td>
</tr>
<tr>
<td>DATE</td>
<td>DATETIMETZ</td>
<td>Fully supported.</td>
</tr>
<tr>
<td>DATETIME</td>
<td>DATETIMETZ</td>
<td>Fully supported.</td>
</tr>
<tr>
<td>DATETIME-TZ</td>
<td>DATETIMETZ</td>
<td>Supported for passing the date and the time part.</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>DECIMAL</td>
<td>Supported for numbers with a maximum of 50 digits with up to 10 digits after the decimal point.</td>
</tr>
<tr>
<td>HANDLE</td>
<td>Handle</td>
<td>Fully supported.</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>DECIMAL</td>
<td>Fully supported.</td>
</tr>
<tr>
<td>INT64</td>
<td>INT64</td>
<td>Fully supported.</td>
</tr>
<tr>
<td></td>
<td>DECIMAL</td>
<td>Fully supported.</td>
</tr>
<tr>
<td>INTEGER</td>
<td>INTEGER</td>
<td>Fully supported.</td>
</tr>
<tr>
<td></td>
<td>DECIMAL</td>
<td>Fully supported.</td>
</tr>
<tr>
<td></td>
<td>INT64</td>
<td>Fully supported.</td>
</tr>
<tr>
<td>LOGICAL</td>
<td>LOGICAL</td>
<td>Fully supported.</td>
</tr>
<tr>
<td>MEMPTR</td>
<td>Memptr</td>
<td>Fully supported.</td>
</tr>
<tr>
<td>RAW</td>
<td>Raw</td>
<td>Fully supported.</td>
</tr>
<tr>
<td>ROWID</td>
<td>Rowid</td>
<td>Fully supported.</td>
</tr>
</tbody>
</table>
The UNKNOWN value is not supported for some dataslot types.
Sonic ESB Managed Adapter

The Sonic ESB Adapter enables you to integrate your Business Process Server application with the Sonic ESB (Enterprise Service Bus) solution.

For details, see the following topics:

- Sonic ESB Adapter features
- Overview
- Mapping OpenEdge ABL dataslots to Sonic ESB message
- Working with Sonic ESB Adapter

Sonic ESB Adapter features

The Sonic ESB adapter provides the following features:

- Interacts with the Sonic ESB to invoke a Sonic ESB Process or Service.
- Constructs a JMS message through mapping of OpenEdge ABL dataslots with appropriate Message fields and sends it to a Sonic ESB process, service, or endpoint.
- Provides an option to wait (or not) for a reply from the invoked Sonic ESB process or service or SonicMQ message.

Note: Sonic ESB Adapter support is provided for Sonic 8.0.0, Sonic 8.5 and Sonic 2013 for Sonic Adapter in OE BPM.
Prerequisites

The Sonic ESB adapter requires the following prerequisites:

• Business Process Server version 11.3 or above

• OpenEdge BPM Studio version 11.3 (or above) or Business Process Modeler version 11.3 (or above)

• Sonic version 8.0.0

Overview

The Sonic ESB adapter fulfills the following requirements:

• Invoke a Sonic ESB Process or Service by sending a JMS Message to its Entry Endpoint. If either a process or service is selected for invocation, only the process or service name is captured at configuration time. At the execution time, the entry endpoint of the process or service is located by invoking client management APIs and the request message is sent to that entry endpoint. The benefit of this approach is if the Sonic ESB Process/Service Entry endpoint is later migrated to different JMS destination, the Sonic ESB adapter continue to function without any change.

• Send a JMS Message to a particular JMS destination (also referred to as Endpoint), which resides on Sonic MQ. The message can be constructed using the datalot values of a BusinessManager process instance. The Sonic ESB adapter workstep can be used to map BusinessManager datalots to the message fields (as discussed in Mapping OpenEdge ABL datalots to Sonic ESB message on page 136). At execution time, the Sonic ESB adapter workstep instance constructs a JMS message from datalot values and then sends the message to either the entry endpoint of the Sonic ESB process (or service) or to the selected JMS Destination.

• The Sonic ESB adapter workstep instance—after sending the message,—may want to wait for the response message and populate the output datalot values. Sonic ESB adapter provides an option to select if the adapter workstep instance should wait for reply or not. If true, the Sonic ESB adapter searches for reply message at predefined OpenEdge-specific JMS destinations (listed below) residing on SonicMQ, in the same order of precedence:

  • OpenEdge.ResponseTopic

  • OpenEdge.ResponseQueue

  The user can select any other JMS destination. The selected JMS destination, on which response message is expected, will be set for "JMSReplyTo" header property on the request message. A request message is the one that gets sent from the adapter and response message is the one that gets received by the adapter.

  If an Exception occurs in the Sonic ESB process (or service) or the request message is rejected, a Fault or Rejected Message is sent to the configured reply JMS destination and the Sonic ESB adapter workstep instance gets suspended.

Mapping OpenEdge ABL datalots to Sonic ESB message

Sonic ESB contains the following types of messages:
• Message: can have properties, but no content.

• Multipart Message: consisting of multiple parts, with each part having its own content ID and content Type. content Type can be any valid MIME type. Each part of the Multipart message can be mapped to CHARACTER dataslots.

• Text Message: The body of a Text message can be mapped to a CHARACTER dataslot.

• XML Message: The body of an XML message can be mapped to a CHARACTER dataslot. The content of any CHARACTER dataslot, copied to the XML message body, must be in XML format.

The following table lists the Sonic property types and the supported OpenEdge ABL dataslot type.

<table>
<thead>
<tr>
<th>Sonic property type</th>
<th>Supported dataslots</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>LOGICAL</td>
</tr>
<tr>
<td>byte&lt;sup&gt;4&lt;/sup&gt;</td>
<td>INTEGER</td>
</tr>
<tr>
<td>short&lt;sup&gt;4&lt;/sup&gt;</td>
<td>INTEGER</td>
</tr>
<tr>
<td>int&lt;sup&gt;4&lt;/sup&gt;</td>
<td>INTEGER</td>
</tr>
<tr>
<td>long</td>
<td>INT64</td>
</tr>
<tr>
<td>float&lt;sup&gt;4&lt;/sup&gt;</td>
<td>OE Decimal type (Double)</td>
</tr>
<tr>
<td>double</td>
<td>OE Decimal type (Double)</td>
</tr>
<tr>
<td>string</td>
<td>CHARACTER</td>
</tr>
</tbody>
</table>

Custom properties

The Sonic ESB adapter request message contains the following custom properties, which are useful for the Sonic process or service.

• **SenderTemplateName** property for the process template name.
• **SenderInstanceId** property for the process instance ID.
• **SenderWorkstepName** property for the adapter workstep name.
• **SenderWorkstepCounter** property for the adapter workstep instance loop counter.

Working with Sonic ESB Adapter

The Sonic ESB adapter workstep instance constructs a Sonic ESB message from the input dataslot values and sends it to the configured Endpoint queue or topic.

You can use Sonic ESB adapter in BP Server applications only. You cannot use it in a Web application.

<sup>4</sup> An exception is thrown if the value of the supported OpenEdge ABL dataslot does not fit the Sonic property type.
To execute the Sonic ESB adapter, you need to perform the following operations:

1. Add and configure the Sonic ESB adapter workstep, as discussed in Configuring the Sonic ESB Adapter on page 138.
2. Define the dataslot mapping, as discussed in Defining the dataslot mapping on page 141.
3. Modify default values (optional), as discussed in Defining the Advanced properties on page 141.

Configuring the Sonic ESB Adapter

You can add and configure the Sonic ESB adapter workstep in your process by performing the following procedures.

To configure Sonic ESB adapter:

1. Click the Assign Participants link in the Tasks pane. Expand the Adapters > Managed > SonicESBAdapters folder and drag the default GenericSonicESBAdapter (or a previously configured Sonic ESB adapter instance) to your Content pane, creating an Sonic ESB adapter workstep.

2. Select the workstep, and then right click and select Properties from the pop up menu to open its Properties view, and click the Configuration tab. This tab includes the Configure button and the Change mapping button for defining the adapter configuration and mapping respectively.

3. Click Configure to open the configurator box of the Sonic ESB Adapter Configurator.

Note: You can also open the Sonic ESB Adapter Configurator using the Managed Adapter Browser (available only from Progress Developer Studio for OpenEdge) functionality. In addition to configuring, you can use the Managed Adapter Browser to perform functions like creating a copy, renaming, deleting, as well as importing and exporting the configuration information. For more information, refer to the "Using the Managed Adapter Browser" section of the OpenEdge Getting Started: Developing BPM Applications with Developer Studio.

4. From the Broker Alias box, specify the alias name of the Sonic execution server. You need to ensure that this broker alias name is added in the OEBS_Home\conf\server-config.xml file. On activation of this adapter workstep, the configured properties for the specified alias are retrieved from the server-config.xml file and used to connect to the Sonic Broker in order to send or receive messages. An example of the alias name entry in the server-config.xml file is provided below:

```xml
<Server name="SonicMachine">
  <Alias>SonicBroker</Alias>
  <Url>tcp://198.168.1.1:2506</Url>
  <SuperUser password="Administrator" encrypted="false">Administrator</SuperUser>
  <Description>Description</Description>
  <Category>category</Category>
  <Properties>
    <Property key="DomainName" value="Domain1"/>
  </Properties>
</Server>
```

Note: On resuming a suspended adapter workstep, the server-config.xml file is refreshed.

5. Use the Domain Configuration properties to connect to the Sonic Domain Manager to fetch list of processes, services, and JMS Endpoints defined in that particular domain. You need to ensure that the Sonic execution server is running when configuring this option.
6. Click **Connect** to connect to the Sonic Domain Manager. Once connected, the **Send To**, **Reply**, and **Request Message** tabs are enabled in the Sonic ESB Adapter Configurator.

### Selecting Sonic ESB address

You can use the **Send To** tab of the Sonic ESB Adapter Configurator to select the Sonic ESB Address (Process, Service, or JMS Endpoint) to be used for connection. You can select any of the following options from the **Send To** tab:

- **Process**: to invoke a Sonic ESB process.
- **Service**: to invoke a Sonic ESB service.
- **JMS Endpoint**: to send a message to the selected endpoint.

Depending on your selection, the table provided is populated with the appropriate type of retrieved data.

---

**Note**: To ensure proper functioning, you need to ensure that at the Adapter execution time, the selected Sonic ESB address (Process, Service, or JMS Endpoint) is available on the Sonic Broker server as pointed by the Broker Alias value provided.

To select Sonic ESB address:

1. From the list of retrieved values in the table, select any entry to display the same in the **ESB Address** read-only box. You can also filter the table entries by entering an appropriate filter text in the **Filter** box.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process</td>
<td>Lists the retrieved processes along with description for each process.</td>
</tr>
<tr>
<td>Service</td>
<td>Lists the retrieved services along with the service type.</td>
</tr>
<tr>
<td>JMS Endpoint</td>
<td>Lists the retrieved endpoints along with type (Queue or Topic).</td>
</tr>
</tbody>
</table>

2. From the **Delivery Mode** drop-down list, select the appropriate message delivery mode. The available options are **PERSISTENT**, **NON_PERSISTENT**, **DISCARDABLE**, **NON_PERSISTENT_ASYNC**, **NON_PERSISTENT_REPLICATED**, and **NON_PERSISTENT_SYNC**. Refer to the Sonic product documentation for information.

### Specifying whether adapter should wait for reply or not

You can use the **Request Message** and **Response Message** tabs (if enabled) in the Sonic ESB Adapter Configurator to construct the request message and response message respectively. The available message types and other properties are the same in both these tabs.

The Sonic ESB adapter provides an option in the Sonic ESB Adapter Configurator to select if the adapter workstep instance should wait for reply or not.
To enable this option:

1. From the Reply tab, select the Wait for Reply check box.

2. If the default reply destination topic, "OpenEdge.ResponseTopic" exists, the same appears as the default value in the Reply To box. If this topic does not exist and the destination queue, "OpenEdge.ResponseQueue" exists, the same appears as the default value in the Reply To box. You can change the reply destination from the provided table, which is populated with all the retrieved JMS destinations.

3. Select any JMS destination entry as the JMS destination for the response message. The selected reply JMS destination is set on the "JMSReplyTo" header property of the request message. You can now use the Response Message tab to construct the response message.

For clarification, consider an example where you want to invoke Process1 process whose entry endpoint is Sample.Q1 and whose exit endpoint, fault endpoint, and rejected message endpoint are all tied to REPLY_TO value. If you now select the Wait for Reply check box and assuming that OpenEdge.ResponseTopic exists, the "JMSReplyTo" header property value of request message is OpenEdge.ResponseTopic. The adapter, after sending request message to Sample.Q1 tries to receive message from OpenEdge.ResponseTopic. The response message can be a valid reply message or a fault message or a rejected message. In case of a rejected message, the adapter workstep is suspended. On receiving valid reply message or a fault message, the adapter updates the output dataslots and gets completed successfully. If the adapter does not receive any reply for 1 minute, it gets suspended.

You can use the Request Message tab to construct the request message that is sent to the selected JMS destination at execution time. You can use the Response Message tab to construct the response message (similar to the construct of the request message) that is received from the exit endpoint, mapped to the REPLY_TO address.

**Constructing message for Sonic ESB**

To construct the message from the respective tabs:

1. From the Message Type drop-down list, select the message type to be constructed. The available options are Message, Multipart Message, Text Message, and XML Message.

2. Use the Properties section to define the message properties. Each property name added to the table must be unique.
   - To add a property, click Add to open the Add Property dialog box. Select the property type as listed in Table 19: Sonic Property type and supported dataslot on page 137.
   - To modify a property, click Modify.
   - To remove a property from the existing list, click Delete.

3. Use the Parts section to define the parts of a Multipart message. The Content ID of each part name added to the table must be unique.
   - To add a part, click Add to open the Add Part dialog box. Select the content type as either text/xml or text/plain.
   - To modify a part, click Modify.
   - To remove a part from the existing list, click Delete.
   - Click Move up or Move down to change the order of the added parts. The order of the parts determines the index value, which starts from zero.

4. Click OK to complete the configuration of the Sonic ESB Adapter. This action also automatically opens the Dataslot Mapping dialog box.
Defining the dataslot mapping

After configuring the Sonic ESB Adapter, you must define the mapping of your process dataslots to the request and response (if defined) messages. The Dataslot Mapping dialog box displays the mapping parameters depending on the selected message type in the Sonic ESB Adapter Configurator dialog box.

The Request Message tab and the Response Message tab (available only if response message is enabled) contains the following tabs:

- **Properties**: Contains the input and output property names, which you can map to Source and Target dataslots respectively (as listed in Table 19: Sonic Property type and supported dataslot on page 137).

  **Note**: For property type "string," if the mapped CHARACTER dataslot value is null, the property is not included in the request message.

- **Body**: Available for Multipart, Text, and XML messages.
  - For Multipart messages, the tabs contain the message parts, which you can map to Source or Target dataslots.
  - For Text and XML messages, the tabs contain the `input.message.body` and `output.message.body` parameters, which you can map to Source and Target dataslots respectively.

You do not need to map all the properties and parts (or body) of request and response messages with a dataslot. Those unmapped parameters do not carry any value for request message and their values are ignored for the response message.

Defining the Advanced properties

In case of synchronous communication, the Sonic ESB Adapter thread, after sending the message, calls the `receive()` method of the reply JMS destination with small timeout value. You can configure this value, along with related values, in the Advanced Adapter Properties dialog box.

**To define advanced properties:**

1. From the Configuration tab of the Sonic ESB Adapter Configurator, click Advanced properties.
2. If required, add the following parameters and set their values appropriately.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Valid values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sonic.sequentialurl.enable</td>
<td>true/false (Default: false)</td>
<td>Set it to true to enable the URL access in sequence. For example, if you provide the URLs as tcp://host1:2610, tcp://host2:2612, then the host1 will always be accessed first, and if it is not available, then host2 will be accessed.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Valid values</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>sonic.faulttolerant.enable</td>
<td>true/false (Default: false)</td>
<td>Set it to true to enable load balancing while accessing Sonic ESB.</td>
</tr>
<tr>
<td>sonic.loadbalancing.enable</td>
<td>true/false (Default: false)</td>
<td>Set it to true to enable fault tolerance while accessing Sonic ESB.</td>
</tr>
</tbody>
</table>

3. Modify the following parameter values.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default/ Maximum value in milliseconds</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>adapter.execution.timeout</td>
<td>60000/12000</td>
<td>The timeout value for the response message, after which the adapter workstep instance gets suspended.</td>
</tr>
<tr>
<td>receive.timeout.for.replyToEndpoint</td>
<td>100/5000</td>
<td>The duration for which the adapter instance thread calls the receive() method on the configured endpoint for the response message. If no message is received in this time period, the adapter thread goes into sleep mode.</td>
</tr>
<tr>
<td>thread.sleep.between.receive</td>
<td>50/5000</td>
<td>The duration of the adapter thread in sleep mode. The response message, received during this time period, is stored on the endpoint and is delivered only on the next receive() call.</td>
</tr>
</tbody>
</table>
The Corticon Adapter enables you to use the capability of Corticon to design, manage and maintain business rules without any coding. Corticon adapter allows BPM users to invoke business rules, written in Corticon Studio and deployed on Corticon server, using a business process. Corticon returns messages to BPM in the form of Corticon Rule Messages and these messages are stored in a table Dataslot that is mapped while configuring Corticon Adapter.

Corticon Adapter is one of the Managed Adapters conforming to the Adapter Configuration and Mapping Framework. For more information regarding the managed adapter framework, see Understanding the Managed Adapter framework on page 19.

For details, see the following topics:

• Working with Corticon Adapter

Working with Corticon Adapter

BPM users should work with both BPM and Corticon to use the features of the Corticon Adapter. BPM developers should export BPM Dataslots(Primitive/Prodataset/TempTable) to an intermediate format that can be used by Corticon. The Corticon developer imports this intermediate format, called BRVD(Business Rules Vocabulary Definition), into Corticon and generates the vocabulary and rules. The Corticon developer then deploys the rules onto the Corticon server.

For details on BRVD export, refer to the Exporting and Importing dataslots section in the OpenEdge ® Getting Started: Developing BPM Applications with Developer Studio.
Configuring the Corticon Adapter

You can configure the Corticon adapter workstep in your BPM process by performing the following procedures.

To configure the Corticon Adapter:

1. On the Diagram tab, click the Assign Participants link in the Tasks pane. Expand the Adapters > Managed > CorticonAdapters folder and drag the default GenericCorticonAdapter (or a previously configured Corticon adapter instance) to your Content pane, creating a Corticon adapter workstep.

2. Double-click the workstep to open its Properties view, and click the Configuration tab. This tab includes the Configure button, the Change mapping button, and the Advanced Properties button for defining the adapter configuration, adapter mapping, and advanced properties respectively.

3. Click Configure to open the configurator box of the Corticon Adapter Configurator.

Note:

If User Authentication is enabled for Corticon Server then user needs to enter valid username and password to load services and versions.

4. Enter the details in the Corticon Adapter Configurator window as per the following table and click OK:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server Alias</td>
<td>Enter the corticon server alias</td>
</tr>
<tr>
<td>Username</td>
<td>Enter the User name (If User authentication is enabled for Corticon Server)</td>
</tr>
<tr>
<td>Password</td>
<td>Enter the Password (If User authentication is enabled for Corticon Server)</td>
</tr>
<tr>
<td>URL</td>
<td>The default URL is http://&lt;enter host name here&gt;:8082/axis.(Authentication is through &quot;http&quot; and a valid host name should be added )</td>
</tr>
<tr>
<td>Name</td>
<td>Select the Decision service name from the Name drop-down list</td>
</tr>
<tr>
<td>Latest version</td>
<td>Select the Decision service version from the Version drop-down list</td>
</tr>
</tbody>
</table>

The Dataslot Mapping window opens.

Note: If the Corticon server is not on the same machine as OpenEdge BPM, then modify the server-config.xml file in $DLC$\OpenEdge\oebpm\server\conf as follows:

```xml
<Server name="corticonserver">
  <Alias>CorticonAlias</Alias>
  <Url>http://localhost:8082/axis/services/Corticon</Url>
  <SuperUser password="administrator" encrypted="false">administrator</SuperUser>
    <Description>Default Server Configuration for Corticon Server.</Description>
  <Category>corticonserver</Category>
</Server>
```
Defining the Dataslot Mapping

After configuring the Corticon Adapter workstep, you must define the mapping of your process dataslots (source and target) to the parameters in the selected Corticon file. The Dataslot Mapping dialog box displays the parameters depending on the input file selected in the Corticon Adapter Configurator dialog box.

To define the dataslot mapping for an Corticon adapter:

1. After adding and configuring the Corticon adapter workstep, double-click the workstep to open its Properties view.
2. From the Configuration tab, click Change mapping to open the Dataslot Mapping dialog box.
3. In the Dataslot Mapping window, select the Dataslot type from the Type drop-down list.
4. Select the appropriate Input and Output formats.
5. From the Rule Messages drop-down list, select the table in which you want to store the Corticon messages and click OK.

The Dataslot types are explained in the following table:

### Table 20: Dataslot Types and explanation

<table>
<thead>
<tr>
<th>Dataslot Type</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primitive</td>
<td>This option will export all dataslots of the types: Character, Longchar, DateTimeTZ, Decimal, Integer, Int64 and Logical. Select the appropriate Dataslots for the Input and Output formats. For Primitive type, by default System dataslots are shown. Out type is disabled.</td>
</tr>
<tr>
<td>Table</td>
<td>If you have selected the Table option in the Type drop-down, select the Table name in the Input and Output drop-down lists.</td>
</tr>
<tr>
<td>Dataset</td>
<td>If you have selected the DataSet option in the Type drop-down, select the appropriate Input and Output DataSet.</td>
</tr>
</tbody>
</table>

**Note:** If Primary Key value is not defined, with in table in Temp table or Dataset, a random value appears in the DataKeyValue Node of the received message. If Primary key is defined, then the Primary key value should be seen in the <DataKeyValue> node of the received message.
SharePoint Managed Adapter

The SharePoint Adapter enables you to drive BPM processes around documents that are added or modified in SharePoint. The SharePoint Adapter provides an interface that allows users to perform the following SharePoint operations without the complexities of SharePoint web services:

- Document - Upload, Download, Delete, Get Metadata, and Set Metadata
- Attachment - Upload, Download, and Delete
- List Item - Create, Delete, Get Metadata, and Set Metadata

SharePoint Adapter is one of the Managed Adapters conforming to the Adapter Configuration and Mapping Framework. For more information regarding the managed adapter framework, see Understanding the Managed Adapter framework on page 19.

For details, see the following topics:

- Working with SharePoint Adapter
Working with SharePoint Adapter

SharePoint Adapter can be used to perform the following actions:

- Download files from Document Library
- Download files from List Item Attachments
- Update/Create files in a Document Library
- Add/Update attachment to a List Item
- Remove files from Document Library
- Remove attachments from List Item
- Get Metadata for a Document or List Item
- Update Metadata for a Document or List Item
- Creating List Item
- Deleting List Item

Configuring the SharePoint Adapter

You can configure the SharePoint adapter workstep in your BPM process by performing the following procedures.

To configure the SharePoint Adapter:

1. On the Diagram tab, click the Assign Participants link in the Tasks pane. Expand the Adapters > Managed > SharePointAdapters folder and drag the default GenericSharePointAdapter (or a previously configured SharePoint adapter instance) to your Content pane, creating a SharePoint adapter workstep.

2. Double-click the workstep to open its Properties view, and click the Configuration tab. This tab includes the Configure button, the Change mapping button, and the Advanced Properties button for defining the adapter configuration, adapter mapping, and advanced properties respectively.

3. Click Configure to open the configurator box of the SharePoint Adapter Configurator.

   **Note:** The SharePoint Adapter configurator box has two tabs- Authentication and Settings.

4. Select the radio button for Auto Configuration or Manual Configuration.

5. Enter the details in the SharePoint Adapter Configurator window as per the following table.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto Configuration</td>
<td></td>
</tr>
</tbody>
</table>
Enter the SharePoint server alias. This field is auto-populated with a default server alias that you can click and modify.

Note: Refer to the server-config.xml file for server alias information. If you do not have access to server-config.xml file, contact your Server Administrator.

**Manual Configuration**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode</td>
<td>From the drop-down list, select the mode of authentication. Select <strong>None</strong> for no security, <strong>NTLM</strong> for SharePoint server secure authentication, and <strong>Basic</strong> for authentication without secure authentication. <strong>NTLM</strong> is selected by default.</td>
</tr>
<tr>
<td>Site URL</td>
<td>Enter the SharePoint Site URL.</td>
</tr>
<tr>
<td>Username</td>
<td>Enter the User name (If User authentication is enabled for SharePoint Server).</td>
</tr>
<tr>
<td>Password</td>
<td>Enter the Password (If User authentication is enabled for SharePoint Server).</td>
</tr>
</tbody>
</table>

Note: The "Always use BPM Performer's SharePoint access credentials" checkbox is for use while invoking SharePoint Adapter as an Adaplet. If this checkbox is selected, SharePoint access credentials that are set in the BPM User Profile page should be used while invoking SharePoint Adapter as an Adaplet.

6. Click the **Settings** tab.

7. From the **Action** drop-down list, select the action that you want to perform.

8. The steps to perform after you select a particular action are detailed in the following table.
<table>
<thead>
<tr>
<th>Action for Document</th>
<th>Steps to perform</th>
</tr>
</thead>
</table>
| **Create**          | 1. Select List Item.  
|                     | 2. In the List Name field, enter a name for the List Item.  
|                     | 3. In the List Item Path field, enter the path of the location where you want to store the List Item.  
|                     | 4. Select ALL or Selective for Metadata. If you selected Selective for Metadata, click Add and edit the items in the Properties pane. Map and Temptable are supported when you select All.  
|                     | 5. Click OK.  
|                     | **Note:** The entries in the Properties pane are editable. To edit a Property, click in the white space and enter a name for the property. |
| **Upload**          | 1. Select Document or Attachment.  
|                     | 2. If you selected Document, provide the Library Name and Location URL (including the file) where you want to upload the document and click OK.  
|                     | 3. If you selected Attachment, enter the List Name and the List Item ID where you want to upload the Attachment and enter the FileName of the attachment that you want to upload and click OK.  
|                     | **Note:** Provide the file name in the FileName field. Do not include File Name in the Location URL while uploading multiple documents. |
| **Download**        | 1. Select Document or Attachment.  
|                     | 2. If you selected Document, provide the Library Name and Location URL (including the file name) from where you want to download the document and click OK.  
|                     | 3. If you selected Attachment, enter the List Name and the List Item ID from where you want to download the Attachment and enter the FileName of the attachment that you want to Download.  
<p>|                     | 4. Click OK. |</p>
<table>
<thead>
<tr>
<th>Action for Document</th>
<th>Steps to perform</th>
</tr>
</thead>
</table>
| Delete              | 1. Select one of the three  
|                     | • Document  
|                     | • Attachment  
|                     | • List Item  
|                     | a. If you selected **Document**, enter the Location URL of the document, including the document name that you want to delete.  
|                     | b. If you selected **Attachment**, enter the **List Name**, the **List Item ID** from where you want to delete the Attachment and enter the **File Name** of the attachment that you want to delete.  
|                     | c. If you selected **List Item**, enter the name of the **List Name** and the **List Item ID** which you want to delete.  
<p>|                     | 2. Click <strong>OK</strong>. |</p>
<table>
<thead>
<tr>
<th>Action for Document</th>
<th>Steps to perform</th>
</tr>
</thead>
</table>
| **Get Metadata**    | 1. Select Document or List Item.  
2. Provide the Location URL of the document for which you want to Get Metadata.  
3. If you selected Document:  
   a. Enter the Location URL of the document.  
   b. Select ALL or Selective for Metadata. If you selected Selective for Metadata, click Add and edit the items in the Properties pane.  
4. If you selected List Item:  
   a. Enter the List Name for which you want to get metadata.  
   b. Enter the corresponding List Item ID.  
   c. Select ALL or Selective for Metadata. If you selected Selective for Metadata, click Add and edit the items in the Properties pane.  
5. Click OK.  
**Note:** The Location URL field can be left blank and selected later during Data slot mapping. The entries in the Properties pane are editable. To edit a Property, click in the white space and enter a name for the property. |
| **Set Metadata**    | |
1. Select **Document** or **List Item**.
2. Provide the **Location URL** of the document for which you want to Set Metadata.
3. If you selected **Document**:
   a. Enter the Location URL of the document.
   b. Select **ALL** or **Selective** for Metadata. If you selected **Selective** for Metadata, click **Add** and edit the items in the **Properties** pane.
4. If you selected **List Item**:
   a. Enter the **List Name** for which you want to set metadata.
   b. Enter the corresponding **List Item ID**.
   c. Select **ALL** or **Selective** for Metadata. If you selected **Selective** for Metadata, click **Add** and edit the items in the **Properties** pane.
5. Click **OK**.

**Note:** The Location URL field can be left blank and selected later during Dataslot mapping. The entries in the Properties pane are editable. To edit a Property, click in the white space and enter a name for the property.

The dataslot **Mapping Window** opens. Dataslot Mapping is explained in Defining the Dataslot Mapping on page 154 section.

**Note:** Please note that ServerAlias, UserName and Password are constant parameters and cannot be mapped to dataslots. The following table provides information about parameters and their corresponding dataslot mapping information.

<table>
<thead>
<tr>
<th>Type</th>
<th>Actions</th>
<th>Parameters that can be mapped to a Dataslot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document</td>
<td>Upload, Download, Delete, Set Metadata, and Get Metadata</td>
<td>Site URL(Authentication tab), Location URL, and Library Name</td>
</tr>
</tbody>
</table>
### Defining the Dataslot Mapping

After configuring the SharePoint Adapter workstep, you must define the mapping of your process dataslots. The **Dataslot Mapping** window has a **Configuration** tab that lists the different sources and the parameters for Dataslot Mapping.

On the configuration tab of the Dataslot Mapping window, based on the selections you made in the Adapter Configuration window, enter the details as per the following table.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name</td>
<td>Enter the name of the file for which you want to perform the action.</td>
</tr>
<tr>
<td>Site URL</td>
<td>Enter the SharePoint server URL or select from the drop down list.</td>
</tr>
<tr>
<td>List Item ID</td>
<td>Enter the SharePoint ID of the list item or select from the drop down list.</td>
</tr>
<tr>
<td>List Name</td>
<td>Enter the List Name or select from the drop down list.</td>
</tr>
<tr>
<td>List Item Path</td>
<td>Enter the sever path to the List Item or select from the drop down list.</td>
</tr>
<tr>
<td>Metadata</td>
<td>Enter the Metadata details or select from the drop down list.</td>
</tr>
<tr>
<td>Document</td>
<td>Enter the document name or select from the drop down list.</td>
</tr>
<tr>
<td>Location URL</td>
<td>Enter the Location URL or select from the drop down list.</td>
</tr>
<tr>
<td>Library Name</td>
<td>Enter the SharePoint library name or select from the drop down list.</td>
</tr>
</tbody>
</table>

### Using SharePoint Adapter as an Adaplet

SharePoint adapter should be configured before invoking it through Adaplet feature. Adaplets contain the same functions that Managed Adapters have, but are more compact—for example, they can be included in a form without adding a separate Adapter workstep or additional dataslots. Using an adaplet enables you to add a configured adapter to a workstep’s presentation form and associate it with a form element.

To invoke the SharePoint Adapter using Adaplet:
1. Create the preconfigured SharePoint adaplet in your process.

2. Add the button, **button1** in your form and add the onClick event to this button.

3. Using Graphical Event Logic (GEL) tool, add the "Invoke adaplet" expression for this event with the and select the adaplet name.

4. Click **Save**.

   **Note:**
   - Multiple files cannot be uploaded to List Item while using Adaplet.
   - While downloading multiple files using Adaplet, a .zip file gets saved to the destination folder.
Web Service Managed Adapter

The Web Service Managed Adapter enables businesses to provide (or publish) their services on the Web and to find (or subscribe to) services of other businesses on the Web. Businesses using Web Services can act as either a service provider that, for example, publishes one of your business applications as a Web service, or as a service requester that helps you to locate required Web services.

For details, see the following topics:

- Overview
- Introducing Web Service Adapter
- Using the Web Service Adapter
- Fault handling

Overview

Web Services are means for integrating diverse and geographically distributed applications into a robust and manageable system. It consists of loosely coupled elements, each running on a different platform and using different technology, but sharing a common set of standards that allows them to communicate effectively.
Figure 19: Web Services Interactions on page 158 shows the main participants in a Web Service, and the interactions between them.

Figure 19: Web Services Interactions

The primary participants are the Service Provider and the Service Requestor (Client). When a service is invoked, the Client sends a request to the Provider and receives a response. The communication happens using SOAP (Simple Object Access Protocol) messages — the XML standard for communication between two computer systems.

WSDL (Web Services Description Language) is an XML-based standard for describing a Web Service. The WSDL description specifies the exact format of the request and response messages, details about the communication protocols that are to be used to transfer those messages, and other information about the service. Before a Web Service can be invoked, the client must obtain the WSDL description about the service, usually from the service provider.

The Provider may also choose to advertise the Web Services it has by registering them with a Discovery Agency. A Discovery Agency is the Web Service equivalent of an Internet search engine — it allows a client to search among the published Web Services and to find information about each particular service. A Discovery Agency uses the UDDI (Universal Description, Discovery and Integration) standard to store and provide access to the list of published Web Services.

General information about Web Services can be obtained from Apache Axis (http://ws.apache.org/axis).

Note: For information regarding Web services used in Business Process Server, refer to the Web Services Developer’s Guide.

Introducing Web Service Adapter

The Web Service Adapter allows you to invoke Web services. The adapter can be configured and used in an entirely point-and-click fashion, without having to write any program code.

Figure 20: Web Service Adapter example

A company has automated its order shipping process. Figure 20: Web Service Adapter example on page 158 displays a portion of the "Shipping" application. After an inventory check, the address of delivery must be validated before the order is packaged and shipped. Another company provides an address verification Web Service.
The "CheckAddress" workstep is handled by the Web Service managed adapter, who generates the SOAP request, sends it to the postal address verification service and receives the response, containing the address status and validated address.

Note: In previous releases, the integration with adapters from the iWay Software Universal Adapter Suite was done through an iWay Managed Adapter that we provided. Currently, iWay Software provides a Web Service layer for interaction with their adapter suites. As a result, we recommend that you access iWay Adapters through the Web Service Managed Adapter. For more information about creating iWay adapters through Web Services, refer to the iWay Software documentation at: http://www.iwaysoftware.com/products/webservicesadapters.

Using the Web Service Adapter

As with all our other provided managed adapters, using the Web Service Managed Adapter is a two-stage process. After the Adapter workstep is defined, the adapter is first configured—selecting a particular Web Service that are invoked, and configuring any extra settings that might be needed when invoking the service. In the "Shipping" application example, shown in Figure 20: Web Service Adapter example on page 158, this step corresponds to selecting the provider for the address verification service and specifying which service must be invoked (it is quite possible that the company provides more than one service).

When the configuration is completed, the adapter inputs and outputs for the particular setting are mapped to dataslots or constants, depending on the particular business process where the adapter is used. For example, the address verification service may have one input, "PostalAddress" and one output, "Status", that is either "true" if the address is valid, or "false" if not. The mapping step involves mapping the "PostalAddress" input to the "ADDRESS" dataslot from the "Shipping" application, and the "Status" output to the "ADDR_VALID" LOGICAL dataslot.

Configuring the Web Service Adapter

To configure Web Service Adapter in your process, follow the steps:

1. Open the process template diagram for which you have to add the Web service adapter.
2. From the Tasks pane, click the Assign Participants link.
3. Expand Adapters > Managed > Web ServiceAdapters.
4. Drag the adapter icon to the Content pane. It is displayed as a Web Services Adapter workstep.
5. Double-click the workstep to open its Properties view.
6. From the Configuration tab, click Configure. The Web Service Adapter Configurator dialog box appears.

Note: You can also open the WebService Adapter Configurator using the new Managed Adapter Browser (available only from Progress Developer Studio for OpenEdge) functionality. In addition to configuring, you can use the Managed Adapter Browser to perform functions like creating a copy, renaming, deleting, as well as importing and exporting the configuration information. For more information, refer to the "Using the Managed Adapter Browser" section the OpenEdge Getting Started: Developing BPM Applications with Developer Studio.

The configurator dialog has two main areas: Web Service Configuration and WSDL Detail.
In the Web Service Configuration area, you can enter the URL of the WSDL file, containing the Web Service definition. For WSDL URL, you can paste the URL or type it directly. You can also use the UDDI Browser to search and select a service, by clicking Search UDDI. The use of the UDDI Browser is covered in Using the UDDI Browser on page 160 below.

If obtaining the WSDL requires authentication or you want to define other security settings, expand the Security Settings panel. For more information on defining security settings, see Defining the security settings on page 160.

Using the UDDI Browser

Use the default UDDI Browser to locate a valid WSDL file in a UDDI registry.

To open the UDDI Browser and locate a WSDL file in a UDDI registry, follow the steps:

1. Click Search UDDI in the Web Service Adapter Configurator dialog box. The UDDI Browser dialog appears.
2. Specify your search criteria.
3. Select a UDDI registry from those listed in the UDDI Registry URL drop-down list.
4. Enter criteria in the Search For box. If you enter * or %, the UDDI Browser finds all available services.
5. Select ways to sort your results in the Sort results by drop-down list. Options in this list include the following:
   - Alphabetical (Ascending)
   - Alphabetical (Descending)
   - Date (Ascending)
   - Date (Descending)

   **Note:** If either of the date options is selected in the above list, the search result sorts the list of services on basis of the published date.

6. Select the maximum number of results you seek from the drop-down list.
7. Select the Exact match results only checkbox to obtain only exact matches to your search criteria.
8. Select the Case sensitive search checkbox to search for specified case sensitive criteria.
9. Click Find. The Messages dialog box appears and states the number of services that match your search criteria.
10. Click OK to view matching WSDL files in the Search Results area on the right side of the UDDI Browser.
11. Valid WSDL files are indicated by an enabled font; that is, the file name is not greyed out. Select a valid WSDL file and information about it is displayed below.
12. Click Select. The URL address of the WSDL file appears in the WSDL: box in the Web Service Adapter Configurator dialog box.

Defining the security settings

The Web Service adapter uses Apache WSS4J to provide support for Web Service security specifications that include: UserName Tokens, Timestamps, SAML Tokens, Digital signatures, and Message encryption.

Expand the Security Settings panel to display all the Security Settings features. By default, this panel is open, and the User Name and Password boxes are blank. However, if the Web service you are invoking requires your authentication to access the service or for run-time http authentication, enter the User Name and Password to your Web server in the WSDL HTTP Authentication section or the Runtime HTTP Authentication section.
You can also choose one of the following options for run-time http authentication:

- **UserName Token:** You must provide a `username.wsdd` file that is available in the system classpath.

- **SAML Token:** SAML, or Security Assertion Markup Language, is a XML-based framework that ensures communications transmitted over Web Services are secure. You must provide a SAML file with a `*.wsdd` extension that is available in the system classpath.

If the Web Service you invoke expects SAML Tokens, provide the following client WSSD (Web Service Deployment Descriptor), which meets Apache WSS4J requirements. This `client-deploy.wsdd` file has the following format:

```xml
<service name="STPing">
  <requestFlow>
    <handler type="java:org.apache.ws.axis.security.WSdoAllSender">
      <parameter name="action" value="Timestamp SAMLTokenUnsigned"/>
      <parameter name="samlPropFile" value="keys/saml.properties"/>
    </handler>
  </requestFlow>
</service>
```

The name of the `client-deploy.wsdd` file must be unique across the Business Process Server installation and available in the application server classpath. A `OEBPS_HOME\WebService\config` folder is added to the system classpath of the Portal and EJB servers. All `*.wsdd` files are maintained in this folder.

- **Other:** For advanced users who want to make use of SAML, digital signatures, and encryption. You must use an actual file name with a `*.wsdd` extension that is compliant with wss4j requirements. This `*.wsdd` file must be available in the system classpath.

- **To provide the details of the web service from the server-config.xml, select the Read from server-config.xml checkbox and enter the server alias in the ServerAlias text box.** Using this checkbox, the users can enable reading the Authentication details and Endpoint URL from the server-config.xml

**Note:**

The server-config.xml should have configuration properties in the below mentioned format:

```xml
<Server name="WebServiceServer">
  <Alias>WebServiceAlias</Alias>
  <Url>http://localhost:8080/WebServiceExample/user</Url>
  <SuperUser password="Password" encrypted="false">UserName</SuperUser>
  <Description>Default Server Configuration for WebService Server.</Description>
  <Category>webserviceserver</Category>
  <Properties>
    <Property key="AuthenticationType" value="UserNameToken/SamlToken/Other/HTTP/NTLM" />
    <Property key="Value" value="path to token file" />
  </Properties>
</Server>
```

In the above example, the two parameters **AuthenticationType** and **Value** are used for different types of authentication. The following table provides the different authentication types and their corresponding values:

<table>
<thead>
<tr>
<th>Authentication Type</th>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserName Token</td>
<td>UserNameToken</td>
<td>Path to username.wsdd file</td>
</tr>
<tr>
<td>SAML Token</td>
<td>SamlToken</td>
<td>Path to saml file with <code>*.wsdd</code> extension</td>
</tr>
</tbody>
</table>
Supporting passing of session tokens to Web Services

If a Web Service is protected by SiteMinder or a similar type of security and requires you to pass cookie information in addition to the security measures of entering your user name and password, the Web Service Adapter provides session token support. By default, the session support is disabled and hidden.

Follow the steps to enable session token support, as follows:

1. Configure the Web Service Managed Adapter workstep (without session token), and click **OK** to close the Map Configurator.
2. Save the project.
3. From the Window menu, select **Show View > Other > Navigator**. The Navigator pane appears on the left.
4. Right-click the project node in the Navigator pane and select **Build Project**. The maps directory appears in the Navigator pane. You may have to refresh the Navigator pane to see the directory.
5. From the **maps** directory, select the workstep for which you want to enable session cookies and open the **config.xml** file.
6. Change the value of the "SESSION_TOKEN_SUPPORT" parameter to "true", as shown in bold in the following code sample:

```xml
<configform synchronous="true" version="1.2">
  <block title="Inputs" description="Inputs" hidden="false">
    <param name="SESSION_TOKEN_SUPPORT" access="enable" description="SESSION_TOKEN_SUPPORT" type="java.lang.String" value="true"/>
  </block>
</configform>
```

7. **Save** and **close** **config.xml**.

Setting session token cookies from dataslot

Follow the steps to set session token cookie from dataslot as follows:

1. Open the Adapter Configurator again. (After performing the preceding procedure.)
When the session support is enabled, the Set session token cookie from datalot checkbox will appear in the Advanced tab of the Web Service Adapter Configurator.

2. Select the Set session token cookie from datalot check box if your Web services use SiteMinder to provide authentication. If you select this checkbox, an additional input (named "session_token") will be provided for the Web service.

3. Map this input to a dalot that contains the SiteMinder session token.

**Reviewing information on WSDL details**

After you enter the WSDL Location, either manually or using the UDDI Browser, in the WSDL: box, click Go to load the WSDL and display information about the services in the "WSDL Detail" area.

In order to illustrate the operation of the adapter configurator, we use the sample "MathService" service, available at the following URL:


**Follow the steps to understand a sample service:**

1. Enter the MathService URL in the **WSDL field**.

2. Click **Go**. The relevant Web Service information is displayed.

   In the left panel, MathService has the "Add" arithmetic operations.

3. Click the Add tree node to update the Information tab to show relevant information.

   Part of the information, displayed when an operation is selected, is the list of operations inputs and outputs, as well as the data type, associated with each input or output. Once the adapter is configured, those parameters become inputs and outputs to the adapter and can be mapped to dalots.

**Using the Web Services Adapter Configurator dialog box**

The adapter configurator provide three other tabs in addition to the Information tab. The Types and Mapping tab is used to define mapping for complex and/or custom data types, and is described in Dealing with complex data types on page 165. The Advanced tab is used to define timeout parameters (see Using the Advanced tab on page 164). The Test tab, where a configuration can be tested actually invoking the service, is described in the following section.
Open the Test tab on the Web Service Adapter Configurator.

**Figure 21: Web Service Adapter Configurator - Test tab**

The Test tab also lists all the inputs and SOAP Response outputs, and lets you specify values for the input parameters. It also displays a box where for the "Add" operation of MathService, there are two inputs, "num1" and "num2", both integer numbers. The single output is named "AddResult", and also is an integer. Note that you can also enter Array type inputs, in the format java.lang.Integer[2; 3; 4].

### Verifying the Add web service

**Follow the steps to verify the Add service example:**

1. Click Reset to clear the values in the Test tab.
2. In the Value column, enter numerical values for the 'num1' and 'num2' inputs.
3. Click Invoke to invoke the operation and it passes the values you specified as parameters. The response message contains the sum of the "num1" and "num2" values, in the "AddResult" parameter.

**Note:** This example only applies to simple data. For Web Services where the outputs are complex types see the example in Figure 22: Web Service Adapter Configurator - Types and Mapping tab on page 165. For cases where both inputs and outputs are complex data, testing is not supported. For more information, see Dealing with complex data types on page 165. For information on using XML output, see Using XML output for complex data types on page 168.

### Using the Advanced tab

Open the Advanced tab to define timeout settings. If required, you can enter the number of seconds you want the workstep to wait to complete the web services invocation. If the defined timeout is exceeded, the workstep is suspended. By default, the timeout is set to "-1", or for an infinite time.
The Advanced tab also can enable you to provide session token support. By default, the session support is disabled and hidden. If you want to enable it, see the procedures in Supporting passing of session tokens to Web Services on page 162.

Dealing with complex data types

Most Web Services have input and output parameters that are of basic data types, such as String, Integer, or Decimal. Some services, however, use complex data types to represent entities such as employee information, product order, and so on.

If the Web Service you are going to use does not contain complex types, you may skip this section and continue to Defining the dataslot mapping on page 170.

When a complex object is either passed as an input parameter to a Web Service, or is returned from the service as an output, the mapping between the CHARACTER type and the corresponding Java class must be defined in the Types and Mapping tab.

Follow the steps to understand a sample service:

2. Click Go. The following screen is displayed.

Note: Regardless of the public Web Service you use, the following section describes how to use complex data types in the Web Service Adapter.

Figure 22: Web Service Adapter Configurator - Types and Mapping tab

The Web Service Adapter Configurator makes its best effort to simplify your work, and it automatically defines the type mapping for every complex type it encounters. The automatically generated definition assumes the following:

- Java class name is the same as the XML type localpart name
- Java class is a JavaBean type object, whose fields can be manipulated with a set of set/get methods
Generating Java classes for Web Service parameters

When dealing with complex data types (as discussed in Dealing with complex data types on page 165), you can use the Web Service adapter to generate Java classes with the serializer (and deserializer) information. You must use a Common resource project with the Web Service adapter, in order to generate the Java classes.

To generate Java class using a Common Resource project:

1. Select the Common Resource project in the Project Explorer view, then click **Tools > Managed Adapters**. The Managed Adapter Browser tool appears.
2. Create a copy of the generic WebService adapter. For the copied Web service adapter, click **Configure**. The **Web Service Adapter Configurator** dialog box appears.
3. Enter the WSDL URL, shown in Figure 22: Web Service Adapter Configurator - Types and Mapping tab on page 165, then click **Go**.
4. For the GetInfo operation, click the **Types and Mapping** tab (see Figure 22: Web Service Adapter Configurator - Types and Mapping tab on page 165).
   a) Select the **Java Input and Output** option.
   b) Click **Generate classes** (enabled only for Common Resource projects). The **Generate classes** dialog box appears.
   c) Modify the package name, if required. Select the **Generate helper classes for serialization** checkbox to generate the serializer helper classes.
   d) Click **OK**. On refreshing the common resource project, the generated classes are available in the Business Objects folder.
5. You can use the generated classes as an object dataslot in the BP Server or Web application. To enable this, you must add the Common Resource project as a dependent project to the BP Server or Web application.
   a) In the Project Explorer view, select the project to which you want to add the Common Resource project.
   b) From **Project** menu, click **Properties**, then click **Java Build Path > Projects**.
   c) Add the common resource project to the build path, then click **OK**.
6. In the above BP Server (or Web application) project, create the Object dataslot, which can be used as input (or output) to the Web service.
7. Publish the projects to the Business Process Server, in order to make the Object dataslot classes available at runtime.
8. Perform this step only if you are using JBoss (Embedded or Enterprise version) application server. After publishing the projects to the Business Process Server, the ClassNotFound exception occurs at runtime as the published bean classes cannot locate the axis classes.
   a) Stop the Business Process Server, if running.
   b) Create a JAR file (example, bo.jar) containing all the classes generated by the Common Resource project. Copy the JAR file to the OEBPS_Home\lib folder.
   c) Delete the common resource classes. published in the OEBPS_Home\ebmsapps\common\bo\classes folder.
   d) Edit and save the OEBPS_Home\jboss\server\ejbServer\conf\jboss-service.xml file as follows:
      - Add the JAR file name with comma (in this case, bo.jar,) as a new line after the axis.jar, line.
   e) Start the BP Server and reinstall the BP Server (or Web application) project containing the Object dataslot.
Using Java input and XML output

The following steps guide you to generate XML output from Java input:

1. Select the Java Input and Soap Message Output option.
2. Click Edit. The Mapping Editor dialog box appears. Modify the type mapping definition, only if any of the preceding assumptions is not true.
3. Provide the required information and Click OK.

Use the Mapping Editor to enter values (or edit the already present values) for the XML-type, fully qualified class name for the Java object that represents the data type. Sometimes a complex object type may contain another complex object as one of its fields. In this case, you may have to manually define the XML mapping for the nested type(s). If needed, you can also define your custom serializer and deserializer factory that are used to convert the Java object to XML form and back.

Generally, when using a Web Service, the provider can give you enough information about what each of these fields must contain. Additional information about writing custom serialization and deserialization factories can be obtained from Apache Axis (http://ws.apache.org/axis/).

**Note:** For more information about complex data types, see the BP Server examples topic in Progress OpenEdge business Process Server: Managed Adapters Guide or use the example provided under OEBPS_HOME\BP Server\examples\WebserviceComplexType.zip.

Finally, note that Web Services containing complex type inputs cannot be tested through the Web Services Adapter Configurator. In this case, make sure to properly test your configuration by running the process from Business Process Portal.

**Note:** The classes for the complex types are not reloadable. If there is any change in the binary, stop the application Server and restart them.

Using XML Input and Output

The WebService managed adapter can construct the input XML on the fly by using the XML format plugin and mapping dataslots to certain XML elements. Use Soap Message as Input and Output option to have input and output in XML format.

**To use XML Input and Output follow the steps:**

1. Select Soap Message as Input and Output option, in the Types and Mapping tab.
2. Open the Test tab. The XML input is displayed on the SOAP Request text area.
3. Check Map Request fields. The Request Mapper option is available.
4. Click Request Mapper. The XPath Wizard appears.
5. Click Add to add a parameter. The Parameter Editor dialog appears.
   a) Enter the parameter name in the Name box.
   b) Browse and select the X path and Click OK.
   c) Select the type from the Type drop-down list.
   d) Click OK to return to the XPath wizard.
   e) Click OK to return to the WebService Adapter Configurator.
6. Provide the values and click Invoke. The XML output appears in the lower text area.

7. Click OK to complete the configuration, and open the Dataslot Mapping dialog window (see Defining the dataslot mapping on page 170).

**Using XML output for complex data types**

In case the Web Service you are using contains complex output and you do not want to use a custom Java class to handle it (see Dealing with complex data types on page 165), you can select the Java Input and Soap Message Output option that is displayed in the Types and Mapping tab and shown in Figure 22: Web Service Adapter Configurator - Types and Mapping tab on page 165.

Selecting the Java Input and Soap Message Output option indicates that you are expecting the result to be returned as an XML string and not as a Java object. All the complex data type outputs will disappear from the mapping table, as seen in Figure 22: Web Service Adapter Configurator - Types and Mapping tab on page 165.

If you use the Java Input and Soap Message Output option, all result data will be returned as a single output that, when you are mapping dataslots (see Defining the dataslot mapping on page 170), you can map to a CHARACTER dataslot. In some situations, returning XML may not be convenient or necessary; for instance, if you are interested in only a few values that are contained in the XML response. In this case, you can use the built-in XPath Editor to directly return the data of interest, instead of the complete response XML.

**To use XML output for complex data types:**

1. Select Java Input and Soap Message Output option in the Types and Mapping tab.
2. Open the Test tab.
3. Add values for each of the Java inputs listed.
4. Click Invoke, and the XML output appears in the lower text area, as shown in the following figure.

   **Figure 23: Web Service Adapter Configurator - Test Tab -Java Input**

   ![Web Service Adapter Configurator - Test Tab -Java Input](image)

   The Map Response fields option is available.

5. Check **Map Respond fields**. The Response Mapper option is available.
6. Click **Respond Mapper**. The **XPath Wizard** appears.

7. Click **Add** to add a parameter. The **Parameter Editor** dialog appears.
   a) Enter the parameter name in the **Name** box.
   b) Browse and select the X path and Click **OK**.
   c) Select the type from the **Type** drop-down list.

8. Click **OK** to complete the configuration, and open the Dataslot Mapping dialog box (see Defining the dataslot mapping on page 170).

For information on the XML Plugin, see **Using the XML Format plug-in** on page 83. Also, note that you cannot use XML plugin internal variables with the Web Service Adapter (for more information on use of variables, see the Using variables and conditions in XPath topic in Progress OpenEdge Business Process Modeler: User’s Guide).

When using the XML plugin, all the selected fields are available for mapping to dataslots as adapter outputs. The response XML is not available as an output.

### About proxy support

If your web service is using a proxy server, you must add the proxy information:

- as a JVM option, if testing needs to be done, or
- to the EJB server or Portal server if the Web Service managed adapter needs to be executed in a BP Server or a Web application.

For more information on how to add the proxy-related information in the JVM, use Axis documentation at the following URL: [http://ws.apache.org/axis/java/client-side-axis.html#NetworkConfiguration%3F](http://ws.apache.org/axis/java/client-side-axis.html#NetworkConfiguration%3F).

If you are using Progress Developer Studio for OpenEdge, you must modify the shortcut for starting Progress Developer Studio for OpenEdge (with Eclipse as IDE) in order to pass the arguments to the JVM. To do so, list the JVM arguments after a "-vmargs" argument, as shown in the following example:

```
-vmargs-Dhttp.proxyHost=my.proxy.server.com
-Dhttp.proxyPort=8080
-Dhttp.proxyUser=user1
-Dhttp.proxyPassword=pass1
```

**Note:** If the proxy information is added to Progress Developer Studio for OpenEdge, then the WSDL should be available in the local file system. When loading the WSDL from the filesystem, use the `file:///` notation, for example, `file:///C:/data/MyService.wsdl`.

### Invoking the Web service Managed Adapter using SSL

Occasionally, a Web service needs to be invoked that uses https to handle secure transactions. When configuring the adapter, the WSDL must be available in the local file system, as is the case for proxy support.

For runtime, you must add JVM parameters for testing the Web service or in the Portal server or EJB server JVM for the run time inside the BP Server Applications or a Web Applications.

For the client side, you must add the following JVM parameters for SSL communication with Java and Tomcat:

- `-Djavax.net.ssl.trustStore=C:/myFolder/myKeyStoreFile // place where the keystore is located`
- `-Djavax.net.ssl.trustStorePassword=changeit // trust store password`
For more information on Tomcat, SSL, and Java, refer to the following Web sites:
http://tomcat.apache.org/tomcat-5.5-doc/ssl-howto.html

Defining the dataslot mapping

Once the Web Service Managed Adapter is configured, the Web Service inputs and outputs must be mapped
to the dataslots that are present in your particular business process.

The Map Configurator dialog box appears automatically when the adapter’s configuration is complete. You can
also manually invoke the Map Configurator by clicking Change Mapping… in the Configuration tab of the
Properties view for the Web Service Adapter workstep.

The appearance of the Map Configurator dialog box depends on the inputs and outputs of your web service.
The following sections describes each of the available tabs.

Using the Inputs tab

The Inputs tab includes the list of inputs for the web service.

For the MathService example (see Configuring the Web Service Adapter on page 159), the Map Configurator
lists the "A" and "B" input parameters in its "Inputs" tab.

From the combo box beside each input parameter, select the dataslot that you want to be associated to the
input. In this case, the value for the A input comes from the dataslot X, and the B input receives its value from
the dataslot Y. You can also directly enter constant values in the combo box for each input — in this case, the
given is used for each invocation of the adapter, and you don’t need to define a separate dataslot for it.

Using the Outputs tab

The Outputs tab containsthe list of the output parameters from the Web Service, and allows you to specify
target dataslots for each value. In the MathService case, the Outputs tab contains the "AddResult" parameter,
containing the sum of the A and B parameters.

Here, we mapped the AddResult so the value from the addition is stored in the Z dataslot.

Notice that the dataslot data types are not mentioned anywhere in the mapper. In case the type of a dataslot
does not match the type of an input or output, the managed adapter framework converts the value appropriately.
Not all such conversions may make sense in the context of a given Web Service or Business Process Server
process, so be careful when mapping different types, and consult the information in the "Supported Types for
Conversion" table in the "Developing Custom Managed Adapters" chapter in the Customization Guide.

Using the Advanced tab

The Advanced tab enables you to map the Web Service results at run time to another machine (Target Endpoint),
if required. Enter the Target Endpoint Address (that is, the URL address of the WSDL Soap address location)
in the Source combo box, or select a dataslot that contains the value of the Target Endpoint Address location.

If you do not know the Target Endpoint Address, you can find it in the service name, WorkflowWSService. The
wsdlsoap:address parameter is shown in bold in the following sample code.

```
- <wsdl:service name="WorkflowWSService">
  - <wsdl:port binding="impl:BPServer1SoapBinding" name="BPServer1">
    <wsdlsoap:address location="http://10.1.5.169:18793/sbm/services/BPServer1" />
  </wsdl:port>
</wsdl:service>
</wsdl:definitions
```
Fault handling

The WebService manager adapter execution gets suspended when it receives an exception while invoking the operation in the target web service. Some of the exceptions thrown by the target web service might have already been described in the WSDL as Faults for that operation. These faults are equivalent to the checked exceptions in Java. Based on the needs of the application, the application might want to catch the fault and take the necessary action.

You can map the fault to a dataslot, as described in the Data slot mapping for fault handling on page 172. You can find out if an exception is occurred or not, by checking the values of the dataslots configured for faults. If there is no exception, the dataslots mapped to the faults will not be set.

All the other exceptions not described as faults suspends the adapter execution.

Describing a fault

Each operation in WSDL can have one (or more) declared faults. As in Java, you can declare the exceptions that are thrown, using the "throws" keyword.

Example, public Product getProduct() throws ProductNotFoundException;

An example of a method with fault handling is provided in WSDL as shown below.

```xml
<wsdl:operation name="getProduct" parameterOrder="in0">
    <wsdl:input message="impl:getProductRequest" name="getProductRequest"></wsdl:input>
    <wsdl:output message="impl:getProductResponse" name="getProductResponse"></wsdl:output>
    <wsdl:fault message="impl:ProductNotFoundException" name="ProductNotFoundException"></wsdl:fault>
</wsdl:operation>
```

In this example, the method converts the "ProductNotFoundException" to a fault in the SOAP operation.

The following sections describe the implementation of the SOAP fault handling in the Web Service managed adapter configurator.

Fault handling in Adapter Configurator

The Web Service Configurator reads the fault descriptions from WSDL and uses the same in configuring the adapter.

The WSDL Detail section (see Reviewing information on WSDL details on page 163) in the Web Service configurator displays the fault handling information in each of the following tabs:

- **Information tab**: Displays fault information such as Operation Info, Input Parameters, Return Parameters, and Faults.

- **Types and Mapping tab**: Click the Java Input and Soap Message Output or Java Input and Output option in this tab to display all the fault types to map to the Java Exception classes. If required, you can change the name of the Exception java class.

For Common Resource projects, you can click Generate classes to generate the classes for this project. If the selected operation has any fault handling defined for it, the corresponding exception classes can be generated. Each of the generated exception class extends the "java.lang.Exception" class.
When generating the classes, you can select if you want helper classes for serialization and deserialization process to be generated or not. It is not mandatory to generate the helper classes.

- **Test tab**: This tab now displays the SOAP Fault tab in addition to the SOAP Response tab. The SOAP Response tab is activated if the result of the web service is successful, else the SOAP Fault tab is activated. The SOAP Fault tab displays content only if there is an exception.

The Input section renders the content based on the inputs required for the SOAP service.

- For primitive type input, this section displays a grid with the Java type, parameter name, and editable value as its columns.
- For complex type input, this section displays the SOAP XML in an editable text area, where you can set the values of particular XML elements as input.

The output of the Web service execution is displayed either in the SOAP Response tab or SOAP Fault tab. The content rendered in these tabs follows the same behaviour as in the Input section. As fault types and exception classes are of complex types, the SOAP Fault tab displays the SOAP fault XML in a read-only text area.

You can select the Map Fault Fields checkbox in the SOAP Fault tab to map an XPath expression in the SOAP fault XML to a user-defined parameter. This checkbox is enabled only if you have clicked either the Soap Message as Input and Output or Java Input and Soap Message Output options in the Types and Mapping tab.

**Mapping an XPath expression to a user-defined parameter**

**To map XPath expression in the SOAP fault XML to a user-defined parameter:**

1. Provide an invalid input to the Web service to generate a fault.
2. Click Invoke to activate the SOAP Fault tab with the SOAP fault XML.
3. Select the Map Fault Fields checkbox, then click Fault Mapper to launch the XPath Wizard.
4. Click Add to specify the parameter name, XPath expression, and the type.
5. Click OK will add the parameter to the XPath Wizard. You can create multiple parameters to map to multiple xpath in the SOAP faults.

---

**Note**: In case multiple fault handlers are defined for this operation, you need to perform the above procedure for each of them.

**Dataslot mapping for fault handling**

In order to implement fault handling, you need to map the defined fault parameters to dataslots. You can define this mapping in the Dataslot Mapping dialog box, which now displays the additional Faults tab.

The parameter displayed in the Faults tab depends on the option you have selected in the Types and Mapping tab (Figure 22: Web Service Adapter Configurator - Types and Mapping tab on page 165) of the Web Service Configurator.

- For Java Input and Output

In this case, the Faults tab displays the fault names and parameters, which you can map to exception objects.

- For Soap Message as Input and Output or Java Input and Soap Message Output
• In case you have not selected the Map Fault Field check box to provide the XPath expression, the Faults tab displays the SOAPFaultXML parameter, which you can map to a CHARACTER dataslot.

• In case you have selected the Map Fault Field check box to provide the XPath expression, the Faults tab displays all the parameters defined in the XPath wizard, which you can map to different dataslots.
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.

**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 datalsots. 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.