# Table of Contents

## Preface

- About this manual ............................................................... 7
- Typographical conventions .................................................. 7

## Chapter 1: Introducing OpenEdge Business Rules

- Concepts .................................................................................. 10
- Terminology related to OpenEdge Business Rules ...................... 10
- Tasks ....................................................................................... 12
  - Integrating Corticon Studio and Progress Developer Studio for OpenEdge ........................................ 12
  - Configuring Corticon Server on the OE Web Server ........................................................................ 12
  - Generating Business Rules Vocabulary Definition ........................................................................ 14
  - Using Outline view to generate business rules vocabulary definition .............................................. 15
  - Exporting Business Rules Vocabulary Definition ........................................................................... 15
  - Setting Business Rules preferences ................................................................................................. 16
  - Using Business Rules API to invoke Decision Services .................................................................. 17
  - Adding Business Rules libraries to PROPATH ................................................................................. 17
- Reference .................................................................................. 18
  - Export Business Rules Vocabulary Definition wizard ................................................................. 18
  - Generate Business Rules Vocabulary Definition wizard .................................................................. 19
  - OpenEdge Business Rules API reference ....................................................................................... 19
For details, see the following topics:

- About this manual
- Typographical conventions

About this manual

This manual provides the Progress® Developer Studio for OpenEdge® online help as a PDF file. This PDF file contains the same information as the Progress Developers Studio for OpenEdge online help. However, the online help is preferable with regard to navigation and formatting. If you are running Progress Developers Studio for OpenEdge, you can access the online help from the Help menu.

Typographical conventions

This documentation uses the following typographical and syntax conventions:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bold</strong></td>
<td>Bold typeface indicates commands or characters the user types, provides emphasis, or the names of user interface elements.</td>
</tr>
<tr>
<td><em>Italic</em></td>
<td>Italic typeface indicates the title of a document, or signifies new terms.</td>
</tr>
<tr>
<td>SMALL, BOLD CAPITAL LETTERS</td>
<td>Small, bold capital letters indicate OpenEdge key functions and generic keyboard keys; for example, <strong>GET</strong> and <strong>CTRL</strong>.</td>
</tr>
<tr>
<td>Convention</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| KEY1+KEY2          | A plus sign between key names indicates a **simultaneous** key sequence: you press and hold down the first key while pressing the second key. For example, `CTRL+X`.
| KEY1 KEY2          | A space between key names indicates a **sequential** key sequence: you press and release the first key, then press another key. For example, `ESCAPE H`.

**Syntax:**

<table>
<thead>
<tr>
<th>Fixed width</th>
<th>A fixed-width font is used in syntax, code examples, system output, and file names.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed-width italics</td>
<td>Fixed-width italics indicate variables in syntax.</td>
</tr>
<tr>
<td>Fixed-width bold</td>
<td>Fixed-width bold italic indicates variables in syntax with special emphasis.</td>
</tr>
<tr>
<td>UPPERCASE fixed width</td>
<td>ABL keywords in syntax and code examples are almost always shown in upper case. Although shown in uppercase, you can type ABL keywords in either uppercase or lowercase in a procedure or class.</td>
</tr>
<tr>
<td>Period (.) or colon (:)</td>
<td>All statements except <code>DO</code>, <code>FOR</code>, <code>FUNCTION</code>, <code>PROCEDURE</code>, and <code>REPEAT</code> end with a period. <code>DO</code>, <code>FOR</code>, <code>FUNCTION</code>, <code>PROCEDURE</code>, and <code>REPEAT</code> statements can end with either a period or a colon.</td>
</tr>
<tr>
<td>[]</td>
<td>Large brackets indicate the items within them are optional.</td>
</tr>
<tr>
<td>[ ]</td>
<td>Large brackets are part of ABL.</td>
</tr>
<tr>
<td>{}</td>
<td>Large braces indicate the items within them are required. They are used to simplify complex syntax diagrams.</td>
</tr>
<tr>
<td>{}</td>
<td>Small braces are part of ABL. For example, a called external procedure must use braces when referencing arguments passed by a calling procedure.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>Ellipses indicate repetition: you can choose one or more of the preceding items.</td>
</tr>
</tbody>
</table>
Introducing OpenEdge Business Rules

Progress Developer Studio for OpenEdge and Progress Corticon Studio are standalone Eclipse-based development environments that can be integrated into a single Eclipse instance to use the capabilities of integrated business rules in Progress OpenEdge. Progress Corticon is a Business Rules Management System. Its patented "no-coding" rules engine automates sophisticated decision processes. It provides the simple yet powerful Corticon Decision Services Methodology for modeling business rules.

Integrating OpenEdge and Corticon enables you to use ABL data structures (such as ProDataSets and temp-tables) as Corticon data structures (Vocabularies and entities). OpenEdge business rules provide tooling support for creating and updating Corticon Vocabularies, and runtime support for simple invocation of Corticon Decision Services.
You can use the integrated environment of Progress Developer Studio for OpenEdge and Corticon Studio to do the following:

- Use a Business Rules server to perform some or all of the business logic from an ABL client
- Re-use the existing business logic data structures (ProDataSets and/or temp-tables) with a Business Rules server without modification
- Generate output from a Business Rules server easily, using ABL data structures, instead of XML
- Generate a resource containing ProDataSet and temp-table schema information from an ABL source code
- Create ABL source code from a resource containing ProDataSet and temp-table schema information
- Import Corticon Vocabulary from a resource containing ProDataSet or temp-table schema information
- Export Corticon Vocabulary to a resource
- Test an OpenEdge application containing both ABL and Corticon business logic from within Progress Developer Studio for OpenEdge
- Create a single project in Progress Developer Studio for OpenEdge containing both ABL and Corticon artifacts

For details, see the following topics:

- Concepts
- Tasks
- Reference

**Concepts**

**Terminology related to OpenEdge Business Rules**

The following terms describe the various components that apply to an integrated development environment of OpenEdge and Corticon:

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProDataSet</td>
<td>A predefined view of data usually from multiple data sources (such as temp and database buffers) that are related to each other through data relationships. It is a potentially complex in-memory data structure that can be passed as a parameter with a single handle from one procedure to another, within a single OpenEdge session or between sessions.</td>
</tr>
<tr>
<td>Component</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Vocabulary</td>
<td>A structured dictionary containing all necessary business terms and relationships between them used by the Business Rules. It is an eCore model that defines a relational data structure against which rules are applied. The relational data structure contains domains, entities, attributes and relationships (including cardinality), and maps to the ABL file's ProDataSet temp-tables. A Vocabulary is used to build rule models in a Rulesheet or tests in a Ruletest.</td>
</tr>
<tr>
<td>Rulesheet</td>
<td>A set of rules (conditions and actions and plain language statements) written from a common business Vocabulary. By organizing these rules, it becomes a self-contained, independent unit of automated decision-making.</td>
</tr>
<tr>
<td>Ruleflow</td>
<td>A set of one or more Rulesheets organized for sequential execution. A Ruleflow aggregates and organizes Rulesheets into a single unit of automated decision-making. It may be assembled from multiple Rulesheets, provided that the Rulesheets use the same Vocabulary file. In other words, a Ruleflow can only have one associated Vocabulary. After a Ruleflow has been saved and deployed to the Corticon Server, it is called a Decision Service.</td>
</tr>
<tr>
<td>Ruletest</td>
<td>A mechanism within Corticon Studio for creating use cases or test scenarios of sample data and sending them to a Rulesheet or Ruleflow for processing. Ruletests consist of one or more Testsheets which test independent Rulesheets, Ruleflows, or can be linked together to test a succession of Rulesheets or Ruleflows to simulate a process sequence. Like Rulesheets, Ruletests also use a common Vocabulary model.</td>
</tr>
<tr>
<td>Decision Service</td>
<td>Automates a discrete decision-making task. A Decision Service is implemented as a set of Business Rules and exposed as a web service or a Java service. By definition, the rules within a Decision Service are complete and unambiguous. For a given set of inputs, a Decision Service addresses every logical possibility uniquely, ensuring decision integrity. After they are deployed to the Corticon server, Ruleflows become Decision Services. Multiple versions of a single Decision Services can be deployed concurrently.</td>
</tr>
<tr>
<td>Business Rules server</td>
<td>A high-performance and scalable system resource that manages pools of Decision Services and executes their rules against incoming requests. It can be easily configured as a Web Services server that exposes Decision Services as true Web Services.</td>
</tr>
</tbody>
</table>

**See also**

Introducing OpenEdge Business Rules on page 9
Tasks

Integrating Corticon Studio and Progress Developer Studio for OpenEdge

You can use the capabilities of Business Rules in Progress OpenEdge by integrating Progress Developer Studio for OpenEdge and Progress Corticon Studio into a single environment.

To integrate Corticon Studio plugins to Progress Developer Studio for OpenEdge, use one of the following methods:

- While installing Corticon Studio, select the Eclipse location of Progress Developer Studio for OpenEdge as the external Eclipse location on the **External Eclipse configuration** page.
- After installing Progress Developer Studio for OpenEdge and Corticon Studio, open **Command Prompt**, go to the installation directory of Corticon Studio and run the following code:

  integrateCorticon.bat -install <PDSOE Eclipse location>

**Note:** For information on integrating Progress Developer Studio for OpenEdge plugins to Corticon Studio, see the *Progress Corticon Documentation*.

See also

[Introducing OpenEdge Business Rules](#) on page 9

Configuring Corticon Server on the OE Web Server

OpenEdge is shipped with the OE Web Server. The Tomcat server is available in your OpenEdge installation directory `%DLC%/servers/tomcat`. It provides a Java container and a web server.

You must configure the Corticon server on your Tomcat server to make it available for deploying projects.

By default, Corticon Server is a part of the Corticon Studio installation but has the following license restrictions:

- The request must come from the same machine on which the server is running. Requests from any other machine or host are disallowed.
- Only a single reactor is available per Decision Service. This means that only one Decision Service can be executed at a time. This allows you to deploy multiple Decision Services simultaneously. Since ABL is single-threaded and since this license is intended for a single-user development scenario, it is ideal for the intended use case. If you want to test the system more thoroughly, you must purchase a proper licensed server.

**Note:** These conditions only apply to the execution of a Decision Service. They do not affect the service availability check and other administrator functions.

To configure the Corticon server:

1. Based on the Corticon Server (purchased or default) that you are using, do one of the following:
• If you are using the default Corticon server that comes with the Corticon Studio installation, copy the Axis.war file located in the Corticon Studio installation directory to the OpenEdge %DLC%\servers\tomcat\webapps directory.

• If you have purchased the Corticon server license, download the Corticon Server for Java Archive from the Progress Download Center available at http://www.progress.com/esd, and copy the Axis.war file located in it to the OpenEdge %DLC%\servers\tomcat\webapps directory.

2. Create a batch file in the OpenEdge work directory (%DLC%\WRK) and specify its name.

3. Open the batch file and set the value of the JRE_HOME variable as: SET JRE_HOME=%DLC%\jdk\jre

4. Start the OpenEdge Web Server.
   a) Open the Servers view in an OpenEdge perspective of Progress Developer Studio for OpenEdge by selecting Window > Show View > Other > Server > Servers > OK.
   b) Select the REST Manager instance and then, select Start in the context menu or on the Servers toolbar.

   **Note:** You can also start the OpenEdge Web Server from the Proenv command prompt using protc start.

5. In a web browser, enter the URL for the OpenEdge Web Server with default ports: http://localhost:8980/axis, and go to the Corticon Server console to verify if the Corticon Server is successfully installed.

   The **Corticon Server Console** login page appears after successful installation. The Corticon server has the following six defined server console login user names:

<table>
<thead>
<tr>
<th>User name</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>admin</td>
<td>admin</td>
</tr>
<tr>
<td>administrator</td>
<td>changeme</td>
</tr>
<tr>
<td>modeler2</td>
<td>modeler2</td>
</tr>
<tr>
<td>modeler1</td>
<td>modeler1</td>
</tr>
<tr>
<td>Tester</td>
<td>tester</td>
</tr>
</tbody>
</table>

   **Note:** See the **Progress Corticon Documentation** for more information about using the Corticon server and for other installation options.

**Security**

Securing the OE Web Server is part of the process of configuring the Corticon Server, especially if it runs within the OE Web Server. Since Corticon Server runs as an independent WebApp, you can secure access to the Rules Service using the security models available in the OE Web Server. For more information on securing the OE Web Server, see the **OpenEdge Development: Mobile Applications book**.
Generating Business Rules Vocabulary Definition

A Business Rules Vocabulary Definition (.brvd) file is used to exchange ProDataSet and/or temp-table schema information between OpenEdge and Corticon at design time. It is a proprietary format of an XML-based file. The schema generated and exported from OpenEdge can be imported and used as the basis for Vocabulary entities and attributes in Corticon Studio. For more information on importing business rules vocabulary definition, see the Progress Corticon Documentation.

To generate a Business Rules Vocabulary Definition file:

1. Create an OpenEdge project or select an existing one.
2. From the project or editor's context (right-click) menu, select Progress OpenEdge > Generate Business Rules Vocabulary Definition.

   **Note:** You can also select the Generate Business Rules Vocabulary Definition menu option from the Outline view. See Using Outline view to generate business rules vocabulary definition for more information.

   The Generate Business Rules Vocabulary Definition wizard appears.

3. In the left tab, select the ABL source file whose ProDataSets and/or temp-tables you want to add to the Business Rules Vocabulary Definition file.

4. By default, the wizard displays the Workspace Resources tab that lists files in the workspace. To view files in the file system, select the drop-down arrow and then select File System.

   **Note:** You can use the Browse ABL Files option available on the toolbar of the File System Resources tab to add files from the local file system to the current list. This option is enabled only when you select File System. To use database temp-tables containing the LIKE keyword you must select a workspace resource.

5. When you select a file in the left tab, the ProDataSets and Temp-Tables tab displays all the ProDataSets and temp-tables available in that file. Select the ProDataSet and/or temp-table that you want to include in the Business Rules Vocabulary Definition file.

   **Note:** If the selected file has any compilation errors, then no ProDataSets and temp-tables are displayed for that file.

   The Selected data structures list displays the ProDataSets and/or temp-tables that you select in the ProDataSets and Temp-Tables tab. To remove an item from this list, select the item and then select on the Selected data structures toolbar.

6. In the Definition file field, click Browse to specify a location in your file system and a name for the Business Rules Vocabulary Definition file.
Note: If you have specified a default rules Vocabulary folder on the Business Rules preferences page, the location of that folder is displayed when you click Browse. You can change this location if necessary. You can also specify a workspace location for the Business Rules Vocabulary Definition file. In which case, you must refresh the OpenEdge project containing the file to be able to view it in the Project Explorer view.

7. Click Finish.

A Business Rules Vocabulary Definition (.brvd) file is created in the specified location with the selected ProDataSets and/or temp-tables. If the same file already exists, a dialog prompts you to replace it.

See also
Exporting Business Rules Vocabulary Definition on page 15
Generate Business Rules Vocabulary Definition wizard on page 19

Using Outline view to generate business rules vocabulary definition

You can use the Outline view to open the Business Rules Vocabulary Definition wizard to generate a Business Rules Vocabulary Definition file.

1. Open the procedure file in the Project Explorer view.
2. Select Window > Show View > Outline.
3. Select a ProDataset or a TempTable, right-click and select Generate Business Rules Vocabulary Definition.

For more information on generating business rules Vocabulary definition, see Generating Business Rules Vocabulary Definition on page 14.

Exporting Business Rules Vocabulary Definition

A Business Rules Vocabulary Definition (.brvd) file is used to exchange ProDataSet and/or temp-table schema information between OpenEdge and Corticon at design time. It is a proprietary format of an XML-based file.

The schema generated and exported from OpenEdge can be imported and used as the basis for Vocabulary entities and attributes in Corticon Studio. For more information on importing business rules vocabulary definition, see the Progress Corticon Documentation.

To export a Business Rules Vocabulary Definition file:
1. Select File > Export > Progress OpenEdge > Business Rules Vocabulary Definition.

Note: You can also select this option from the context (right-click) menu in the Project Explorer view.

The Export Business Rules Vocabulary Definition wizard appears.
2. In the left tab, select the ABL source file whose ProDataSets and/or temp-tables you want to add to the Business Rules Vocabulary Definition file.
3. By default, the wizard displays the Workspace Resources tab that lists files in the workspace. To view files in the file system, select the drop-down arrow and then select File System.
Note: You can use the Browse ABL Files option available on the toolbar of the File System Resources tab to add files from the local file system to the current list. This option is enabled only when you select File System. To use database temp-tables containing the LIKE keyword you must select a workspace resource.

4. When you select a file in the left tab, the ProDataSets and Temp-Tables tab displays all the ProDataSets and temp-tables available in that file. Select the ProDataSet and/or temp-table that you want to include in the Business Rules Vocabulary Definition file.

Note: If the selected file has any compilation errors, then no ProDataSets and temp-tables are displayed for that file.

The Selected data structures list displays the ProDataSets and/or temp-tables that you select in the ProDataSets and Temp-Tables tab. To remove an item from this list, select the item and then select on the Selected data structures toolbar.

5. In the Definition file field, click Browse to specify a location in your file system and a name for the Business Rules Vocabulary Definition file.

Note: If you have specified a default rules Vocabulary folder on the Business Rules preferences page, the location of that folder is displayed when you click Browse. You can change this location if necessary. You can also specify a workspace location for the Business Rules Vocabulary Definition file. In which case, you must refresh the OpenEdge project containing the file to be able to view it in the Project Explorer view.

6. Click Finish.

The selected ProDataSets and/or temp-tables are exported to the specified Business Rules Vocabulary Definition file.

See also
Generating Business Rules Vocabulary Definition on page 14
Export Business Rules Vocabulary Definition wizard on page 18

Setting Business Rules preferences

The Business Rules preferences page allows you to specify the default folder where the Business Rules Vocabulary Definition (.brvd) file is located.

To set preferences for Business Rules:

1. Select Window > Preferences > Progress OpenEdge > Business Rules.

2. At the Rules vocabulary folder field, select Browse to specify the location where you want to create the Business Rules Vocabulary Definition file.

The location that you specify here becomes the default location of the Business Rules Vocabulary Definition file and is displayed when you select the Browse button at the Definition file field of the Generate Business Rules Vocabulary Definition and Export Business Rules Vocabulary Definition wizards. You can change this location at the time of generating or exporting the Business Rules Vocabulary Definition file.
Using Business Rules API to invoke Decision Services

OpenEdge allows you to use Business Rules API to call the Decision Service that is deployed onto the Corticon Server. A SOAP message is sent to the Corticon Server and Corticon updates the values of the entity and returns them to OpenEdge. The returned Vocabulary model structure is the same as the one passed.

**Note:** To invoke a Decision service from ABL business logic, you must add the required procedure library and include files to the project's PROPATH. See Adding Business Rules libraries to PROPATH for more information.

To invoke a Decision Service, use the following API classes:

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenEdge.BusinessRules.RulesServerConnectionParameters</td>
<td>Returns parameters for connecting to a Business Rules server. Initially, access to Decision Service and Admin Service are through separate connections (although they can be specified by one parameter).</td>
</tr>
<tr>
<td>OpenEdge.BusinessRules.RulesServerConnection</td>
<td>Describes the server connection for an OpenEdge Rules Server. This is a specialized class whose main purpose is to provide an abstraction from the fact that the connection to the Rules server is a Web Services connection.</td>
</tr>
<tr>
<td>Progress.Json.ObjectModel.JsonObject</td>
<td>Denotes a dynamic number of properties, each addressable by a Unicode string called a name.</td>
</tr>
<tr>
<td>Progress.Lang.AppError</td>
<td>Is the ultimate super class of all application errors. An application error is simply any collection of data you need to provide meaningful information about a condition. Representing a user-defined error as an error object allows your application to throw and catch or return the error in the ABL structured error handling model.</td>
</tr>
</tbody>
</table>

Adding Business Rules libraries to PROPATH

You can configure an OpenEdge project with Corticon settings to run Corticon-related functionality from within Progress Developer Studio for OpenEdge. To do so, you must add the Business Rules libraries (OpenEdge.BusinessRules.pl file) to the list of procedure libraries appended to PROPATH in Progress Developer Studio for OpenEdge.

**Note:** You must also add these entries to the PROPATH of any production environments.
To add the Business Rules libraries to PROPATH:

1. Select a project in the **Project Explorer** view and select **Properties** from the context (right-click) menu or select **Project > Properties** from the main menu bar.

   The **Properties** page for the selected project appears.

2. Select **Progress OpenEdge > PROPATH**.

   **Note:** For a shared AVM project, select **Window > Preferences > Progress OpenEdge > Shared AVM > PROPATH**.

3. Select **Add External Library** and select the **OpenEdge.BusinessRules.pl** file from the file system.

   **Note:** If the project uses GUI as the runtime, the **OpenEdge.BusinessRules.pl** file is located in @{DLC}\gui\rules and if the project uses TTY as the runtime, the file is located in @{DLC}\tty\rules, where DLC is the default installation directory.

The selected procedure library appears in the PROPATH section (the left pane on the PROPATH properties page).

4. To use the output of the GetMessages() API (which, returns the messages for the most recent decision service invocation), add the @{DLC}\gui\rules or @{DLC}\tty\rules folder to the PROPATH along with the **OpenEdge.BusinessRules.pl** file by selecting **Add External Directory** on the PROPATH properties page.

5. Click **OK**.

   **Note:** You can also add PROPATH entries while creating a project using the **Define PROPATH** page.

**See also**

*Using Business Rules API to invoke Decision Services* on page 17

**Reference**

**Export Business Rules Vocabulary Definition wizard**

The **Export Business Rules Vocabulary Definition** wizard allows you to export ProDataSets and temp-tables from the workspace and/or file system to an intermediate Business Rules Vocabulary definition (.brvd) file. This intermediate .brvd file can be imported to Corticon Studio to generate and package Vocabulary and rules and then deployed onto the Corticon server.

The **Export Business Rules Vocabulary Definition** wizard includes the following controls:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workspace Resources</td>
<td>(Default) Displays a list of ABL class and procedure files available in the workspace.</td>
</tr>
<tr>
<td>Control</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>File System Resources</td>
<td>Displays a list of files that you select from the file system. This tab appears only when you select File System using the drop-down arrow on the toolbar.</td>
</tr>
<tr>
<td>ProDataSets and Temp-Tables</td>
<td>Displays the available ProDataSets and temp-tables for each selected file.</td>
</tr>
<tr>
<td>Selected data structures</td>
<td>Displays the ProDataSets and temp-tables that you select in the ProDataSets and Temp-Tables tab.</td>
</tr>
<tr>
<td>Definition file</td>
<td>Specifies the location of the Business Rules Vocabulary Definition file to which the selected ProDataSets and temp-tables are exported.</td>
</tr>
</tbody>
</table>

See also
Exporting Business Rules Vocabulary Definition on page 15

Generate Business Rules Vocabulary Definition wizard

The Generate Business Rules Vocabulary Definition wizard allows you to generate a Business Rules Vocabulary Definition (.brvd) file for the selected ProDataSets or temp-tables. This intermediate .brvd file can be imported to Corticon Studio to generate and package Vocabulary and rules and then be deployed onto the Corticon server.

The Generate Business Rules Vocabulary Definition wizard includes the following controls:

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workspace Resources</td>
<td>(Default) Displays a list of ABL class and procedure files available in your current workspace.</td>
</tr>
<tr>
<td>File System Resources</td>
<td>Displays a list of files that you select from the file system. This tab appears only when you select File System using the drop-down arrow on the toolbar.</td>
</tr>
<tr>
<td>ProDataSets and Temp-Tables</td>
<td>Displays the available ProDataSets and temp-tables for each selected file.</td>
</tr>
<tr>
<td>Selected data structures</td>
<td>Displays the ProDataSets and temp-tables that you select in the ProDataSets and Temp-Tables tab.</td>
</tr>
<tr>
<td>Definition file</td>
<td>Specifies the location where the Business Rules Vocabulary Definition file is saved.</td>
</tr>
</tbody>
</table>

See also
Generating Business Rules Vocabulary Definition on page 14

OpenEdge Business Rules API reference

The following sections describe the various APIs supported by the OpenEdge-Corticon framework for invoking Decision Services:
Class and Interface reference

This section contains reference entries that describe the built-in classes and interfaces supported for OpenEdge Business Rules. These classes and interfaces contain types and other artifacts directly pertaining to OpenEdge and Corticon integration. They also contain types and other artifacts for generic handling of server connections and their parameters.

OpenEdge.BusinessRules.DecisionService class

The OpenEdge.BusinessRules.DecisionService class provides the functionality to create a Decision Service instance.

Constructors

The following constructor invokes the currently available version of a Decision Service, as determined by the Corticon Server, specified by the OpenEdge.BusinessRules.RulesServerConnection class:

```java
PUBLIC (pcDecisionServiceName AS CHARACTER, poServer AS
OpenEdge.BusinessRules.RulesServerConnection)
```

The following constructor invokes the specified version of a Decision Service running on the Corticon Server as specified by the OpenEdge.BusinessRules.RulesServerConnection class:

```java
PUBLIC (pcDecisionServiceName AS CHARACTER, pdVersion AS DECIMAL, poServer AS
OpenEdge.BusinessRules.RulesServerConnection)
```

The following constructor invokes the version of a Decision Service running on the Corticon Server that is effective at the date or time provided, as specified by the OpenEdge.BusinessRules.RulesServerConnection class:

```java
PUBLIC (pcDecisionServiceName AS CHARACTER, ptEffectiveOn as DATETIME-TZ, poServer AS
OpenEdge.BusinessRules.RulesServerConnection)
```

Super Class

Progress.Lang.Object class

Public Properties

<table>
<thead>
<tr>
<th>Name property</th>
<th>Version property</th>
<th>EffectiveOn property</th>
</tr>
</thead>
</table>

Protected Property

ServerConnection property

Public Methods

| InvokeService() method | GetMessages() method | IsServiceAvailable() method |
OpenEdge.BusinessRules.RulesServerConnection class

The OpenEdge.BusinessRules.RulesServerConnection class describes the server connection for an OpenEdge Business Rules server. It uses Web Services connections to access the Business Rules server.

Super Class

OpenEdge.Core.ServerConnection.WebServiceConnection class

OpenEdge.BusinessRules.RulesServerConnectionParameters class

The OpenEdge.BusinessRules.RulesServerConnectionParameters class returns parameters for connecting to a Business Rules server. Although the Decision Service and Administration Service can be specified by one parameter, access to them is through separate connections. The initial implementation uses WebServices but the OpenEdge.BusinessRules.DecisionService class abstracts that from a caller. The implementation affects the parameters values passed in.

Constructors

The following constructor creates a JSON object containing valid connection parameters:

```java
PUBLIC (INPUT poOptions AS JsonObject)
```

The following constructor creates a character representation of a JSON object containing valid connection parameters:

```java
PUBLIC (INPUT pcOptions AS CHARACTER)
```

Interface

OpenEdge.Core.ServerConnection.IConnectionParameters interface

Public Properties

<table>
<thead>
<tr>
<th>DecisionServiceConnectionParameters property</th>
<th>AdminServiceConnectionParameters property</th>
<th>FormatMask property</th>
</tr>
</thead>
</table>

Public Method

GetConnectionString() method

OpenEdge.Core.ServerConnection.FormatMaskEnum class

The OpenEdge.Core.ServerConnection.FormatMaskEnum class contains various simple format masks for connection parameters. These format masks can also be represented by individual format provider classes.

Super Class

OpenEdge.Lang.EnumMember class
Public Properties

<table>
<thead>
<tr>
<th>Default property</th>
<th>None property</th>
<th>DashSpace property</th>
<th>ABLConnect property</th>
</tr>
</thead>
<tbody>
<tr>
<td>DoubleDashSpace property</td>
<td>DoubleDashEquals property</td>
<td>NameEquals property</td>
<td>Custom property</td>
</tr>
</tbody>
</table>

**OpenEdge.Core.ServerConnection.IconnectionParameters interface**

The `OpenEdge.Core.ServerConnection.IconnectionParameters` interface returns connection parameters for the Decision Service calls.

**Public Property**

| FormatMask property |

**Public Method**

| GetConnectionString() method |

**See also**

`OpenEdge.BusinessRules.RulesServerConnectionParameters class` on page 21

**OpenEdge.Core.ServerConnection.IServerConnection interface**

The `OpenEdge.Core.ServerConnection.IServerConnection` interface describes the connection to a server (such as a database, an AppServer, or a Web Service).

**Public Properties**

| Server property | ConnectionParameters property | Connected property |

**Public Methods**

| Connect() method | Disconnect() method | CreateServer() method | DestroyServer() method |

**OpenEdge.Core.ServerConnection.WebServiceServerConnection class**

The `OpenEdge.Core.ServerConnection.WebServiceServerConnection` class describes a Web Service connection.

**Constructors**

PUBLIC (INPUT poConnectionParameters AS OpenEdge.Core.ServerConnection.WebServiceServerConnectionParameters)

**Interface**

`OpenEdge.Core.ServerConnection.IServerConnection interface`
OpenEdge.Lang.EnumMember class

The OpenEdge.Lang.EnumMember class is an enumeration definition class. It is an abstract class.

Constructors

The following constructor allows the specification of an integer value and a character name for the enumeration:

```
PROTECTED (piValue as INTEGER INPUT pcName AS CHARACTER)
```

The following constructor allows only the specification of an integer value for the enumeration:

```
PROTECTED (piValue as INTEGER)
```

The following constructor allows only the specification of a character name for the enumeration:

```
PROTECTED (INPUT pcName AS CHARACTER)
```

Public Properties

<table>
<thead>
<tr>
<th>Name property</th>
<th>Value property</th>
</tr>
</thead>
</table>

Public Method

ToString() method

OpenEdge.Lang.WidgetHandle class

The OpenEdge.Lang.WidgetHandle class is a primitive class for widget-handle variables.

Constructors

The following default constructor results in the WidgetHandle object having an unknown value:

```
PUBLIC ()
```

The following constructor initializes the WidgetHandle's value:

```
PUBLIC (phValue AS HANDLE)
```

Super Class

Progress.Lang.Object class
Public Property

Value property

Public Method

ToString() method

Class Properties and Methods reference

This section contains reference entries that describe each built-in class property and method that ABL supports for working with OpenEdge Business Rules and structured error handling. Class properties and methods are mechanisms that allow you to monitor and control the behavior of class-based objects.

ABLConnect property

The ABLConnect property defaults to `OpenEdge.Core.ServerConnection.FormatMaskEnum:DashSpace`.  
**Data type:** `OpenEdge.Core.ServerConnection.FormatMaskEnum`  
**Access:** PUBLIC STATIC  
**Applies to:** `OpenEdge.Core.ServerConnection.FormatMaskEnum` class

AdminServiceConnectionParameters property

The AdminServiceConnectionParameters property contains connection parameters for the Admin Service calls such as `IsServiceAvailable()`.

**Data type:** `OpenEdge.Core.ServerConnection.IConnectionParameters`  
**Access:** PUBLIC  
**Applies to:** `OpenEdge.BusinessRules.RulesServerConnectionParameters` class

Connected property

The Connected property returns `True` if the server is valid and connected. Otherwise, returns `False`.

**Data type:** LOGICAL  
**Access:** PUBLIC  
**Applies to:** `OpenEdge.Core.ServerConnection.IServerConnection` interface

ConnectionParameters property

The ConnectionParameters property specifies the parameters for the connection.

**Data type:** `OpenEdge.Core.ServerConnection.IConnectionParameters`  
**Access:** PUBLIC  
**Applies to:** `OpenEdge.Core.ServerConnection.IServerConnection` interface

Custom property

The Custom property is a custom mask that is too complex for simple substitute-based parsing.

**Data type:** `OpenEdge.Core.ServerConnection.FormatMaskEnum`
Access: PUBLIC STATIC
Applies to: OpenEdge.Core.ServerConnection.FormatMaskEnum class

DashSpace property
The DashSpace property resolves to '-&1 &2'.
Data type: OpenEdge.Core.ServerConnection.FormatMaskEnum
Access: PUBLIC STATIC
Applies to: OpenEdge.Core.ServerConnection.FormatMaskEnum class

DecisionServiceConnectionParameters property
The DecisionServiceConnectionParameters property contains connection parameters for the Decision Service calls.
Data type: OpenEdge.Core.ServerConnection.IConnectionParameters
Access: PUBLIC
Applies to: OpenEdge.BusinessRules.RulesServerConnectionParameters class

Default property
The Default property defaults to OpenEdge.Core.ServerConnection.FormatMaskEnum:DashSpace.
Data type: OpenEdge.Core.ServerConnection.FormatMaskEnum
Access: PUBLIC
Applies to: OpenEdge.Core.ServerConnection.FormatMaskEnum class

DoubleDashEquals property
The DoubleDashEquals property resolves to '--&1=&2'.
Data type: OpenEdge.Core.ServerConnection.FormatMaskEnum
Access: PUBLIC STATIC
Applies to: OpenEdge.Core.ServerConnection.FormatMaskEnum class

DoubleDashSpace property
The DoubleDashSpace property resolves to '--&1 &2'.
Data type: OpenEdge.Core.ServerConnection.FormatMaskEnum
Access: PUBLIC STATIC
Applies to: OpenEdge.Core.ServerConnection.FormatMaskEnum class

EffectiveOn property
The EffectiveOn property is an optional property that specifies the effective date for selecting the Decision Service to be invoked. It defaults to an unknown value, which is the latest or the most recent version.

Note: Either the Version or EffectiveOn property can be used at a time, which you can decide when you call the constructor.
Data type: DATETIME-TZ
Access: PUBLIC
Applies to: OpenEdge.BusinessRules.DecisionService class

FormatMask property
The FormatMask property must always be set to FormatMaskEnum:Custom for WebService and URL connection parameters.
Data type: OpenEdge.Core.ServerConnection.FormatMaskEnum
Access: PUBLIC
Applies to: OpenEdge.BusinessRules.RulesServerConnectionParameters class; OpenEdge.Core.ServerConnection.IconnectionParameters interface

Name property
The Name property specifies the name of the Decision Service to be invoked. This value is mandatory and must be entered as a constructor argument.
Data type: CHARACTER
Access: PUBLIC
Applies to: OpenEdge.BusinessRules.DecisionService class; OpenEdge.Lang.EnumMember class

NameEquals property
The NameEquals property resolves to '&1=&2'.
Data type: OpenEdge.Core.ServerConnection.FormatMaskEnum
Access: PUBLIC STATIC
Applies to: OpenEdge.Core.ServerConnection.FormatMaskEnum class

None property
The None property does not require a default format mask to be applied.
Data type: OpenEdge.Core.ServerConnection.FormatMaskEnum
Access: PUBLIC STATIC
Applies to: OpenEdge.Core.ServerConnection.FormatMaskEnum class

Operation property
The Operation property is extracted from the WebServiceConnectionParameters object passed in to the class.
Data type: CHARACTER
Access: PUBLIC
Applies to: OpenEdge.Core.ServerConnection.WebServiceServerConnection class

PortName property
The PortName property is extracted from the WebServiceConnectionParameters object passed in to the class.
**Data type**: CHARACTER  
**Access**: PUBLIC  
**Applies to**: OpenEdge.Core.ServerConnection.WebServiceServerConnection class

**Server property**  
The Server property refers to the actual server. The object acts as a wrapper for handle-based servers. For OpenEdge.Core.ServerConnection.WebServiceServerConnection class, the Server property is of type OpenEdge.Lang.WidgetHandle.

**Data type**: Progress.Lang.Object class  
**Access**: PUBLIC  
**Applies to**: OpenEdge.Core.ServerConnection.WebServiceServerConnection class

**ServerConnection property**  
The value for the Version property is mandatory and must be entered as constructor argument.

**Data type**: OpenEdge.Rules.RulesServerConnection  
**Access**: PROTECTED  
**Applies to**: OpenEdge.BusinessRules.DecisionService class

**ServiceName property**  
The ServiceName is extracted from the WebServiceConnectionParameters object passed in to the class.

**Data type**: CHARACTER  
**Access**: PUBLIC  
**Applies to**: OpenEdge.Core.ServerConnection.WebServiceServerConnection class

**Value property**  
The Value property is set through a constructor.

**Data type**: INTEGER  
**Access**: PUBLIC STATIC  
**Applies to**: OpenEdge.Lang.EnumMember class; OpenEdge.Lang.WidgetHandle class

**Version property**  
The Version property is an optional property that specifies the version of the Decision Service to be invoked. It defaults to an unknown value, which is the latest or the most recent version.

**Note**: Either the Version or EffectiveOn property can be used at a time, which you can decide when you call the constructor.

**Data type**: DECIMAL  
**Access**: PUBLIC  
**Applies to**: OpenEdge.BusinessRules.DecisionService class
CreateServer() method
The CreateServer() method creates a server object. This method is separated from the Connect() and Disconnect() methods so that a server can be connected and disconnected multiple times.

Return type: VOID
Access: PUBLIC
Applies to: OpenEdge.Core.ServerConnection.IServerConnection interface
Syntax
```csharp
CreateServer()
```

Connect() method
The Connect() method connects to the specified server based on the ConnectionParameters.

Return type: VOID
Access: PUBLIC
Applies to: OpenEdge.Core.ServerConnection.IServerConnection interface
Syntax
```csharp
Connect()
```

DestroyServer() method
The DestroyServer() method deletes a server object. This method is separated from the Connect() and Disconnect() methods so that a server can be connected and disconnected multiple times.

Return type: VOID
Access: PUBLIC
Applies to: OpenEdge.Core.ServerConnection.IServerConnection interface
Syntax
```csharp
DestroyServer()
```

Disconnect() method
The Disconnect() method disconnects from the server if connected.

Return type: VOID
Access: PUBLIC
Applies to: OpenEdge.Core.ServerConnection.IServerConnection interface
Syntax

```plaintext
Disconnect()
```

**GetConnectionString() method**

**Return type:** CHARACTER  
**Access:** PUBLIC  
**Applies to:** OpenEdge.BusinessRules.DecisionService class; OpenEdge.Core.ServerConnection.IConnectionParameters interface

The following version of the method overrides and returns the connection string from the DecisionServiceConnectionParameters property:

```plaintext
GetConnectionString()
```

The following version of the method overrides and returns the connection string from the DecisionServiceConnectionParameters property. If the ConnectionParameters object has a custom format mask, an AppError is returned.

```plaintext
GetConnectionString(INPUT pcFormatMask AS CHARACTER)
```

The following version of the method overrides and returns the connection string from the DecisionServiceConnectionParameters property. If the ConnectionParameters object has a custom format mask, an AppError is returned.

```plaintext
GetConnectionString(INPUT poFormatMask AS FormatMaskEnum)
```

**GetMessages() method**

The `GetMessages()` method returns the messages for the most recent Decision Service invocation. It is an idempotent operation, so a repeated call to this method always returns the same data. The message data is only cleared when a new InvokeService call is made. The RulesMessage table refers to the response data structure used in the InvokeService() call. These references may differ from the input application data structure. The `GetMessages()` method always returns at least one row containing the version of the Decision Service that was used (for cases where an EffectiveOn value or no version was specified). The message has a severity of Info and displays a message text specifying the version of Decision Service that was invoked. The remaining fields contain unknown values.

**Return type:** VOID  
**Access:** PUBLIC  
**Applies to:** OpenEdge.BusinessRules.DecisionService class
Syntax

GetMessages(OUTPUT TABLE RulesMessage)

InvokeService() method

Return type: VOID
Access: PUBLIC
Applies to: OpenEdge.BusinessRules.DecisionService class

The following version of the method accepts the table handle to be passed to the Decision Service. You must call this method BY-REFERENCE for a shallow copy. The contents of this temp-table are removed and replaced by the results of the Decision Service call. The temp-table passed into this method cannot be part of a ProDataSet, otherwise an AppError is raised.

InvokeService(INPUT-OUPUT TABLE-HANDLE phAppData)

The following overloaded version of the method accepts the table handle to be passed to the Decision Service. You must call this method BY-REFERENCE for a shallow copy. The results of the Decision Service call are returned in the output parameter without updating the input data. The temp-table passed into this method cannot be part of a ProDataSet, otherwise an AppError occurs. The output table is a copy of the input table. The intent of this API is to allow callers to use the input data as a before-image for comparison purposes. The RulesMessage table returned by the GetMessages() method contains a field named DataKeyValue.

InvokeService(INPUT TABLE-HANDLE phAppData, OUTPUT TABLE-HANDLE phResponseData)

The following version of the method accepts the ProDataSet handle to be passed to the Decision Service. You must call this method BY-REFERENCE for a shallow copy. The contents of this ProDataSet are removed and replaced by the results of the Decision Service call.

InvokeService(INPUT-OUPUT DATASET-HANDLE phAppData)

The following version of the method specifies the ProDataSet handle to be passed to the Decision Service. You must call this method BY-REFERENCE for a shallow copy. The results of the Decision Service call are returned in the output parameter without updating the input data. The output data set is a copy of the input data set. The intent of this API is to allow callers to use the input data as a before-image for comparison purposes. The RulesMessage table returned by the GetMessages() method contains a field named DataKeyValue that refers to the tables in the response ProDataSet.

InvokeService(INPUT DATASET-HANDLE phAppData, OUTPUT DATASET-HANDLE phResponseData)

IsServiceAvailable() method

The IsServiceAvailable() method determines whether the Decision Service is available or not.
Return type: LOGICAL  
Access: PUBLIC  
Applies to: `OpenEdge.BusinessRules.DecisionService` class  

Syntax

```
IsServiceAvailable()
```

**ToString() method**

If applied to the `OpenEdge.Lang.EnumMember` class, the `ToString()` method returns the Name property. If the Name property is unknown, it returns the Value property.

Return type: CHARACTER  
Access: PUBLIC  
Applies to: `OpenEdge.Lang.EnumMember` class; `OpenEdge.Lang.WidgetHandle` class  

Syntax

```
ToString()
```