Chapter 2
Introduction

PegaRULES Process Commander applications are designed to complement other systems and technologies that you already have in place for doing work. The Process Commander integration services are based on industry-standard technologies for external system integration. Whether you have information in Process Commander that you want to make available to other systems, or there is information in an external system that you need to access from within a Process Commander workflow, you use the integration services features to accomplish your goal.

This chapter contains the following sections:

■ Overview
■ Services
■ Connectors
■ Data Mapping
■ Integration Accelerators
■ Before You Begin
Overview

Process Commander provides integration facilities that support interactions between its applications and external systems, databases, Web sites, e-mail servers, and so on. By using these features, Process Commander can exchange information with many kinds of external systems, including relational databases, Enterprise Information Systems (EIS), e-mail servers, Excel spreadsheets, and even simple text files.

Process Commander interfaces support a wide range of technologies and standards, including HTML, HTTP, Simple Object Access Protocol (SOAP), Microsoft .NET, Sun Microsystems Enterprise JavaBeans™ (EJB), IBM WebSphere MQ® messaging, Java Message Service (JMS), J2EE Connector Architecture (JCA), Business Process Execution Language (BPEL), JSR 94, JSR 168, and others. These interfaces are called integration services.

Integration services enable communications or exchanges between Process Commander applications and external systems, whether an exchange is initiated from a user interface or browser (by a user) or whether an exchange is initiated from a server or a background process. In integration situations, a Process Commander application can act as either the client or the server, communicating with both back-end and front-end applications.

Integration services are grouped into the following categories:

- **Services** — programmatic components that define and implement an interface between an external application that acts as the client and a Process Commander application that acts as the server.

  Services respond to requests or messages sent from external systems to Process Commander. The messages can request data, deliver data, or even deliver a work object on behalf of the external system.

- **Connectors** — programmatic components that implement an interface that is defined by an external system. With connectors, your Process Commander application acts as the client and the external system acts as the server.
Connectors make requests or send messages to external systems, typically as part of a workflow process. Do not confuse connector rules or interfaces with flow connectors, the arrows on a Visio workflow diagram.

In addition, the following integration components support services and connectors:

- Import and export mechanisms, rules that parse incoming text or files, and rules that assemble messages or requests to be sent to external systems
- Data mapping and transformation functions
- Accelerators that generate rules and interfaces, automating certain integration tasks
- Testing and troubleshooting tools, especially the System Management client application

A Process Commander application typically has multiple connectors and services that make requests to and process requests from external systems.
Services

External systems can make requests to Process Commander applications; services respond to those requests (Figure 2-1).

For example, suppose a Process Commander application contains data about sales taxes in the USA and an activity that calculates the tax. An accounts payable program located elsewhere in the enterprise needs to use the sales tax rate for individual states on a regular basis. In this case, you would expose the activity as a service to make it accessible to the accounts payable program. After you create the service, the program can make calls to Process Commander when it needs to use the current sales tax rate for a state.

![Diagram showing interaction between Client, Service, External System, and Process Commander.]

Figure 2-1. External Clients Access Rules Through Process Commander Services

To define most services, use the Create Service Rules Accelerator (Service Accelerator) to create the key rules and data objects, which is available from the Wizards section of the Integration page. As the name implies, the accelerator prompts you to enter information about the service that you want to create. Then the accelerator generates the appropriate rules and data objects for the service.

A service package is a data object that describes how the services manage state. For services of type Enterprise JavaBean™ (EJB), COM, CORBA, Java, Microsoft .Net, Portlet, and SOAP, the service package form is also used to generate a deployment file appropriate for that technology.

While the service package is the container for the service, the service rule itself holds information about how the individual interactions with the external system. For example, SOAP service rules govern a SOAP exchange. They parse SOAP requests and assemble SOAP responses.
For more information about services, see Chapter 3, “Services Overview.” For information about the Service Accelerator, see Using the Service Accelerator, available on the Pega Developer Network (PDN).
Connectors

Connectors are the mechanism that Process Commander uses to send messages to or make requests of external systems and then process the results or response (Figure 2-2).

For example, suppose that for a task in a Process Commander workflow the user needs the current price of a product. The enterprise has an ERP system that exposes product prices as a Web service and a SOAP connector is used to obtain the appropriate pricing information from the ERP system.

Figure 2-2. A Connector Enables a Workflow to Make Requests to an External Server

To create a connector of type EJB, Java, .NET, SOAP, or SQL, use the Create Connector Rules Accelerator (Connector Accelerator), which is available from the Wizards section of the Integration page. As the name implies, the accelerator prompts you to enter information about the connector that you want to create. Then the accelerator generates the appropriate rules, activities, classes, and properties that the connector needs. (For connectors of type HTTP, JCA, JMS, MQ, and BPEL, you must create the rules manually.)

After you create a connector, use Visio to insert an Integrator task into an existing workflow rule. The Integrator task calls the connector through the activity that the accelerator generated, and passes to the connector parameters from the workflow.

In the example in Figure 2-2, the Integrator task passes in the UPC code of a product to an external SOAP service. As with the service rules, the connector rule holds information about that protocol or technology and, in this case, converts the parameters, their values, and other information into a valid SOAP request and calls
the pricing Web service on the EIS system. The connector parses the returned SOAP response and writes the price into a Process Commander property that the workflow can access.

For more information about connectors, see Chapter 4, “Connectors Overview.” For information about the Connector Accelerator, see Using the Connector Accelerator, available on the Pega Developer Network (PDN).
Data Mapping

When you configure a service or connector, you also match or correlate the properties in the Process Commander application to the corresponding data fields or properties in the external system. This is called **data mapping**. You map both the data that comes in to Process Commander as well as the data sent from Process Commander.

The service or connector can send and receive property values or an HTML or XML stream to and from the external system. When you configure the data mapping for a service or for a connector that is not created by a accelerator, you map the data elements (values) that are used by the service or connector. If incoming data must be parsed before it can be processed — the incoming data is in a delimited file, for example — map the data using a parse rule. If outgoing data needs to be presented as formatted HTML or XML, map the data using an XML or HTML stream rule.

When you use the Service Accelerator or the Connector Accelerator, the accelerator configures the data mapping for the service or connector rule. The Service Accelerator creates suggested lists of input and output parameters based on the purpose of the service. The Connector Accelerator creates input and output lists based on the source material it parses while generating the rules and also creates properties in Process Commander that represent the data in the external system.

Other integration services accelerators also assist you in data mapping by creating class and property rules that implement the data model of an external system. For more information about data mapping, see Chapter 5, “Data Mapping.”
Integration Accelerators

Process Commander provides several rule generation and interface generation accelerators that automate some of the integration tasks. The integration accelerators are accessible from the Wizards section of the Integration page. Additionally, you can use the Publish as Web Service option on the Process tab of a flow rule to convert a flow into a SOAP or BPEL Web service.

Create Connector Rules (Connector Accelerator)

The Connector Accelerator creates connector rules for only those connectors that use a protocol or technology that exposes its metadata. For example, because a SOAP service has a Web Service Definition Language (WSDL) file that defines its interface, the Connector Accelerator can create SOAP connectors.

To use this accelerator, first create an abstract class (typically one that inherits from Data-) for the accelerator to use as the base for the generated rules, classes, and properties for the connector. Then run the accelerator and identify the class name and the file, URL, or database that defines the data structure of the external system.

The accelerator uses the Interface Generator tool to import the metadata of the external system that you want connect to. It then parses and converts the information into data classes and properties (subclasses of the class that you specified), connector rules (instances of Rule-Connect-*), activities that call the connector rules.

You can use the Connector Accelerator to create the following kinds of connectors:

- Enterprise JavaBean™ (EJB)
- Java class
- SQL (relational database)
- SOAP Web service
- Microsoft .NET Web service

For more information, see Using the Connector Accelerator, available on the PDN.
Create Service Rules (Service Accelerator)

The Service Accelerator creates service rules and data objects for several service types. It creates services that do one of the following: create a work item and start the flow for that item, perform a flow action on an existing work item, or call an activity that you have created to do something else.

Run the accelerator, select the type of service you want to create, and fill out the forms that the accelerator displays. Based on your selections, the accelerator creates the appropriate rules and data objects for a service of that type. For services of type SOAP and dotNET, the accelerator also generates a deployment file (a WSDL file). For services of type EJB and Java, you open the generated service package and generate the deployment file manually.

You can use the Service Accelerator to create the following kinds of services:

- EJB
- File
- Java
- JMS
- JSR 94
- MQ
- .NET
- SOAP

For more information, see *Using the Service Accelerator*, available on the PDN.

Import XSD/DTD

The Import XSD/DTD Accelerator implements in Process Commander the data model described in an XSD or DTD file. This accelerator uploads and analyzes a DTD or XSD file that you create or identify and then creates the following according to the options you select:
- Class rules that represent each element or type defined in the document and properties that define each parameter
- Model rules for each class rule
- XML parse rules
- XML stream rules

To use this accelerator, first create an abstract class for the accelerator to use as the base for the generated rules, classes, and properties. Then run the accelerator and make selections from the displayed forms.

For more information, see Data Mapping XML, available on the PDN. See also the Application Developer Help system.

**Import JavaBeans**

Many organizations have pre-existing Java data models, instances of which represent their business or work objects. If requests or responses to/from an external system will include Java objects that are instances of JavaBeans, your applications can use the Java Pages feature to interact with those objects as though they were pages and properties on the clipboard.

You use the Import JavaBeans accelerator when a JavaBean instance will be sent from an external system to a Process Commander Java-based service rule. The accelerator creates class and Java property rules (properties of mode Java Property or Java Property List) that reflect the data model described in a Java class file.

You start the accelerator and identify the source Java class. The accelerator imports the Java class through introspection and then generates the appropriate class and property rules. It also creates Java wrapper classes that implement the get and set methods from the Java class through the PROj ectWrapper interface of the Clipboard Java API. The Java wrapper classes provide you with a standard way to access Java objects as clipboard pages from within Process Commander.
Note: When your Process Commander application needs to call a method in an external Java class, you use the Connector Accelerator to generate a Java connector. If the method invoked by the connector has arguments that are themselves instances of JavaBeans, the Connector Accelerator will generate the appropriate class and property rules and Java wrapper classes in addition to the connector and activity rules.

For more information about using the Java Pages feature, see Working with Java Objects in Process Commander, available on the PDN. For information about the PROobjectWrapper interface, see the PublicAPI JavaDocs topics.

Create EForm Rules (EForm Accelerator)

The EForm Accelerator helps to implement the Process Commander SmartForms feature, which automates the processing of and creation of PDF forms, eliminating manual steps. This accelerator uses a flow to guide you through the process of creating an eForm file rule and its corresponding eForm map rule.

The eForm Accelerator imports and parses a PDF form that you identify. Then it displays a list of the names of the interactive form fields in the PDF form, making it easier for you to match the fields to Process Commander properties. If you create the rules yourself, you must manually enter the name of each form field in an eForm map rule and verify that you made no data entry errors on your own.

Because the accelerator uses a flow rule, the data you enter is stored as a work object until you finish the flow. When you resolve the work object, the eForm Accelerator creates an eForm file rule for the uploaded PDF form and an eForm map rule that specifies how the fields in a form like that one map to Process Commander properties.

For information about using the SmartForms feature to process incoming PDF forms or to generate a PDF form, see Working with PDF Forms and Documents, available on the PDN.
Create External Data Table Rules

The Create External Data Table Rules wizard (External Table wizard) creates rules and data objects that implement in Process Commander the data model of an external database table. With these rules and data objects in place, you can use the Obj-series of activity methods to interact with the data in an external table in the same way that you use them to read and write data to the Process Commander database rather than using a SQL connector rule.

Before you begin, obtain the appropriate JDBC database driver for the database and a database user account for Process Commander. Then configure a data source that points to the external table from the application server Process Commander is running in and create an instance of Data-Admin-DB-Name that uses the data source you configured.

You start the wizard and identify the Data-Admin-DB-Name that you created. The wizard uses the data source to connect to the external database. You select a table, and the wizard generates an instance of Data-Admin-DB-Table, a class rule that represents the table, and property rules that represent the columns in the table. It also generates a model rule for the class rule.

Create Authentication Configuration

The Create Authentication Configuration wizard helps you configure your Process Commander system to use an LDAP-compliant directory server to authenticate users.

The wizard creates an authentication service data object (Data-Admin-AuthService) that holds the connection information for the LDAP directory. It also specifies two standard activities that use the connection information to bind to the directory server, authenticate the users, and re-authenticate users if their sessions expire.

There are two parts to an authentication service: the data object itself and a servlet definition in the Process Commander web.xml file that refers to the data object.

The web.xml file in Process Commander 5.1 has five default servlet definitions: WebStandardLDAP1, WebStandardLDAP2, and so on, up to WebStandardLDAP5.
As long as the name you choose for an authentication service object matches one of the default servlet definitions, you can implement up to five authentication service configurations without having to edit the web.xml file. Note that editing a web.xml file means undeploying and then redeploying Process Commander.

For more information, see *Authentication in PegaRULES Process Commander 5.2*, which is available on the PDN.

**Publish as Web Service Option**

The Publish as Web Service option on the flow forms creates service and connector rules that make a flow available as a BPEL process named PublishedFlows. To use this option, you must first create and save a flow. Then, select the Process tab and click the Publish button.

The Publish feature creates two SOAP service rules and packages them with three SOAP connector rules to create a BPEL Web service. It also generates three WSDL documents and one BPEL document.
Before You Begin

To create a service or connector, you must understand Process Commander terms, forms, functions, and the SmartBuild design and development process. You must have experience in defining Process Commander class rules, properties, models, activities, and other types of rules.

You also need some knowledge of the technology or protocol for which you are building the service or connector. For example:

- To create a SOAP service, you need to know how to write a SOAP client so that you can test the SOAP service.

- To create a SQL connector, you need to know information about the database, such as the database name, database host server, and the JDBC connection pool name so that you can define the database with a Data-Admin-DB-Name data object. Additionally, you need to know how to write SQL queries so that you can test the connector.

- To create Java-based services and connectors, you need to understand how classes are included in the Process Commanders. For information, see PRKB-20931 “About the Process Commander Version 5 Class Paths,” available on the PDN.

- To create a connector, you should also know enough about flow design so that you can attach the connector and test it from within the flow.

For information about related documentation, see the listing of additional publications and resources in Chapter 1, “About This Document.”
RuleSets

If the Process Commander application is designed to use more than one RuleSet, consider how services and connectors interact with the Process Commander system while determining the RuleSet to use for service and connector components:

- Connectors typically run in the current requestor session — for example, the session for the workflow. Assign the RuleSet for connectors to the access groups of the users whose work will trigger connectors from a workflow process.

- Services typically obtain a new, separate requestor ID and session and they use the access group specified in the service package. Assign the RuleSet for services to the access group you specify in the service package.

Additionally, the RuleSet lists of services and connectors and your RuleSet list must include Pega-IntSvcs RuleSet — the standard RuleSet for integration services.

Rule Resolution

Many Process Commander rules have a multipart key that is used to identify the correct rule to use in any situation. The rule that the system uses is based on the key names that you provide when referring to the rule, and on elements such as inheritance, versioning, and security.

The rules and data classes that you create for services and connectors are subject to the same rule resolution algorithms as any other rules or data classes. This means, for example, that you can design multiple versions of the same connector for use in different versions of the same workflow.