Integrating WebSphere Commerce with ILOG JRules for dynamic rules

Lalit Agarwalla
Sanjib K. Mohanty
Jegan Jegadeesan

August 04, 2010

This article addresses how to integrate WebSphere® Commerce with ILOG JRules BRMS, where ILOG BRMS manages the business rules and WebSphere Commerce consumes these rules.

Introduction

WebSphere Commerce is a software platform supporting various business models and business administrative tools, supporting B2C, B2B, or both on a single platform. In various modules within WebSphere Commerce, there are many rules involved for making business decisions. Examples are payment rules, pricing rules, promotion rules, loyalty rules, and so on. The nature of these rules is such that they may change frequently and can be complex and dynamic. This is where WebSphere Commerce can be integrated with ILOG BRMS to manage these business rules. This approach also enables an enterprise-wide central repository of business rules that other applications can also use.

What is ILOG JRules BRMS?

ILOG JRules Business Rules Management Systems (BRMS) is an integrated application development and execution platform that allows organizations to define, deploy, monitor, and maintain a variety of complex decision logic used by operational systems. ILOG BRMS extracts and manages decision logic separately from core application code so that it can be easily understood, maintained, and re-used across the organization.

By externalizing rules from the application code, business experts can define and manage decision logic. This reduces the amount of time and effort required to update decision logic in production systems, and increases the organization’s ability to respond to changes in the business environment.

ILOG JRules BRMS includes the following primary components:

- **Rule Repository**: This is a repository allowing rules to be externalized from the core application code. The repository allows decision logic to be managed as an enterprise asset,
making it easier to understand and update decision logic. It also consolidates decision logic from disparate applications and information silos so that it can be shared and re-used across the organization.

- **Rule Studio and Rule Team Server:** These tools allow business experts to define and manage decision logic that was previously buried in the code. These tools give business functions the ability to define application behavior, while also providing the ability for business and IT to work collaboratively on application development and maintenance.
  - **Rule Studio:** The Rule Studio provides comprehensive rule testing, debugging, and packaging tools for developers. It supports deploying and debugging rule sets to the Rule Execution Server and enables collaboration with business rule authors through integration with the Rule Team Server.
  - **Rule Team Server:** The Rule Team Server is a highly scalable rule management server with a collaborative web environment for authoring, managing, validating, and deploying business rules.

- **Rule Execution Server:** This is a runtime engine that allows production systems to access and execute decision logic managed within the ILOG BRMS. The “rule engine” allows complex and inter-related rules to be executed based on specific business context, using a combination of data inputs, sets of applicable rules, and execution algorithms that define how to process the data and rules in order to provide an output. Figure 1 shows a high level depiction of how a ILOG JRules BRMS system works.

**Figure 1. ILOG JRules BRMS overview**

By adopting the ILOG BRMS approach, organizations can effectively deal with the problems associated with traditional embedded decision logic. The organization gains visibility and access to business rules, along with the ability to more easily define and automate them for use in their operational systems.
Benefits of integrating WebSphere Commerce with ILOG JRules

The ILOG BRMS approach includes the following benefits:

- Provide enterprise-level centralized repository of business rules and policies, instead of WebSphere Commerce having to embed business rules that are not accessible by other applications.
- Access the same rules by other applications like Point of Sales (POS) applications, apart from WebSphere Commerce.
- Reduce and remove a dependency on IT departments for changes in WebSphere Commerce rules.
- Express decision rules with increased precision, using a natural language business vocabulary syntax.
- Change business decision rules by using simple interfaces like Microsoft® Office 2008, which is more familiar to business users.
- Support dynamic complex business rules that might not be supported by packaged applications.
- Test, validate, and simulate the rules before moving into production though ILOG Decision Validation Services (DVS).
- Use rule versioning with sunrise and sunset, along with managing full rules lifecycle.

Integration approaches and architecture

The requirement to integrate WebSphere Commerce and ILOG BRMS is two fold. The obvious aspect is that ILOG BRMS needs to expose the rules as services that can be called by WebSphere Commerce at runtime. The other crucial aspect is that BRMS does not hold any of the application data as it holds just the business rules. These application data reside in WebSphere Commerce and need to be made available to BRMS during the authoring of the rules. Figure 2 shows a simple view of the integration architecture.

Figure 2. WebSphere Commerce and ILOG BRMS integration architecture
There are various ways WebSphere Commerce can expose the application data required for rule authoring. Some of the ways are:

- WebSphere Commerce can expose the data using web services, which BRMS can consume. The advantage is that it is simple to expose the data, whereas you need to have a proper mechanism for sending large sets of data over web services.
- WebSphere Commerce can export the application data. You can import and store this data locally by BRMS in a separate or embedded database. The advantage is that data will be local to BRMS and it can handle the data the way it wants, irrespective of the WebSphere Commerce interface. In this approach, there also needs to be a mechanism to synchronize periodically.

This exposed WebSphere Commerce data will be used while authoring the rules. Once the rules are authored, BRMS can expose the rules using various interfaces like EJB, POJO, and web services. Figure 3 shows a functional diagram of the integration.

**Figure 3. WebSphere Commerce and BRMS functional diagram**

**Authoring rules in JRules by pulling data from WebSphere Commerce**

While authoring rules in JRules, the user needs to be guided on what values to enter. For example, when the user needs to create a rule saying “Give 10 Loyalty points if the product code is FULO-0301”, the system guides the user while typing the product code so that rule authoring errors are reduced. One way to achieve the above example is to display a drop-down with a list of all the eligible product codes and a description for the user to select, as shown in Figure 4.
Figure 4. Sample rule using a guided editor

In Figure 4, the WebSphere Commerce product code (FULO-0101) and product description (Product 1) are populated from WebSphere Commerce. All the required data can be exposed from WebSphere Commerce in various ways as discussed in the above section.

This type of rule authoring is achieved by pulling the desired data from WebSphere Commerce and presenting them in the JRules editors in an intuitive manner. JRules can extend and customize the Rule Studio and Rule Team Server, whereas all the required data from WebSphere Commerce can be displayed.

The next section will briefly describe how to customize the ILOG Rule Studio and Rule Team Server using a custom value editor that enables data from WebSphere Commerce to be displayed for guided authoring. However, elaborate discussion on JRules customization is beyond the scope of the article. For more information, refer to the JRules documentation and samples in the Resources section.

Customizing ILOG Rule Studio authoring

Any customization for ILOG Rule Studio authoring requires development of a custom Eclipse plug-in that will get integrated with the existing Eclipse-based studio environment. These custom extensions are plugged into the Business Object Model (BOM) using the custom properties. Figure 5 shows BOM custom properties for the custom value editor sample. For more information, see the Custom value editor for Rule Studio sample topic.

Figure 5. BOM custom properties for the Value editor

All these mentioned classes need to implement specific extension API interfaces. These classes will typically reside inside the Eclipse plug-in. Also, within the plug-in, you need to write code that fetches the required data from WebSphere Commerce. The plug-in has full control on how the
value editor will be displayed. Java™ swing-based or other appropriate editor boxes need to be developed.

Once the plug-in is developed and deployed, you get functionalities similar to Figure 4.

**Customizing ILOG Rule Team Server authoring**

Customizing the ILOG Rule Team Server is similar to customizing the Rule Studio. It does not involve Eclipse plug-ins, but requires the Team Server EAR file to be repackaged with the customization. Figure 6 shows how the custom value editor is used in the Rule Team Server customization.

**Figure 6. ILOG Team Server custom value editor sample**

For a detailed discussion on how to customize ILOG JRules, refer to the JRules samples and documentation listed in the Resources section.

**Exposing the rules from JRules**

There are various ways (EJB, POJO, or web service) to expose business rules created within JRules BRMS. This section explains how these business rules are exposed using a web service.

This web service is also called a transparent decision service. A transparent decision service is technically a web service with management capabilities. It drives rule execution and enables users to access the Rule Execution Server through a web service, rather than accessing it directly. It is transparent because users do not need to know how it is implemented. They only need to know that it can be accessed through HTTP and XML data formats (SOAP).

An ILOG JRules transparent decision service is either:

- **Hosted**: Hosted Transparent Decision Service (HTDS) is a simple way for you to create a web service when your business rule application is based on XML (with an XML XOM).

- **Monitored**: Monitored Transparent Decision Service (MTDS) is a simple way for you to create a web service when your business rule application is based on a Java XOM.
In the MTDS environment, a separate EAR file is created and deployed in the application server, along with the Rule Execution Server. Once the decision service is deployed, the WSDL will be available in the web service URL. To create an MTDS service and to deploy as a web service, see the Creating a Web service or monitored transparent decision service project for RuleApps topic.

Consuming the rules within WebSphere Commerce

In this section, we will assume that a set of rules are being exposed as a web service by ILOG JRules BRMS and explain how these rules can be consumed by WebSphere Commerce at runtime. WebSphere Commerce provides a development tool in a Rational® Application Developer or Rational Software Architect environment called the WebSphere Commerce Toolkit. The toolkit is used for development.

WebSphere Commerce enhancements for V6 and V7 enable WebSphere Commerce to act as a service consumer. The business application flow in WebSphere Commerce is implemented in the controller command. The controller command in turn uses the task commands to implement the unit business logic. When WebSphere Commerce acts as a service consumer, the client API is called from the task command to send and receive the message using service data objects to the remote system. The client API uses the invocation service to communicate to the remote component, in our case it is the ILOG BRMS. The invocation service internally uses the WebSphere Commerce messaging system to invoke the web service on the external system.

Figure 7 shows the interaction flow from the task command to the external ILOG BRMS.

**Figure 7. Interaction flow**

The implementation steps are:

1. Generate the service data objects from the WSDL exposed by ILOG BRMS.
2. Implement the client API that will use the above SDOs to send the request message and handle response. SDOs are Java data objects for marshalling and unmarshalling of XML data during web services communications.
3. Create a task command to use the client API to make the web services call the external ILOG BRMS.
4. Integrate the task command to the WebSphere Commerce business logic by calling the task command from the appropriate controller command of the business logic flow.
5. Configure the WebSphere Commerce messaging system.

For more information, see the Creating an outbound web service client for WebSphere Commerce topic.
Conclusion

This article discussed how to integrate WebSphere Commerce with ILOG JRules BRMS to provide more robust, dynamic, and complex business rules. It also discussed various integration approaches between WebSphere Commerce and ILOG JRules, and how these rules are exposed by BRMS and consumed by WebSphere Commerce.