Develop decision services, Part 3: Deployment and beyond

Skill Level: Introductory

Raj Rao (rrao2@us.ibm.com)
BRMS IT Specialist
IBM

Sandeep Desai (sandeep@us.ibm.com)
Enterprise IT Architect
IBM

02 Aug 2011

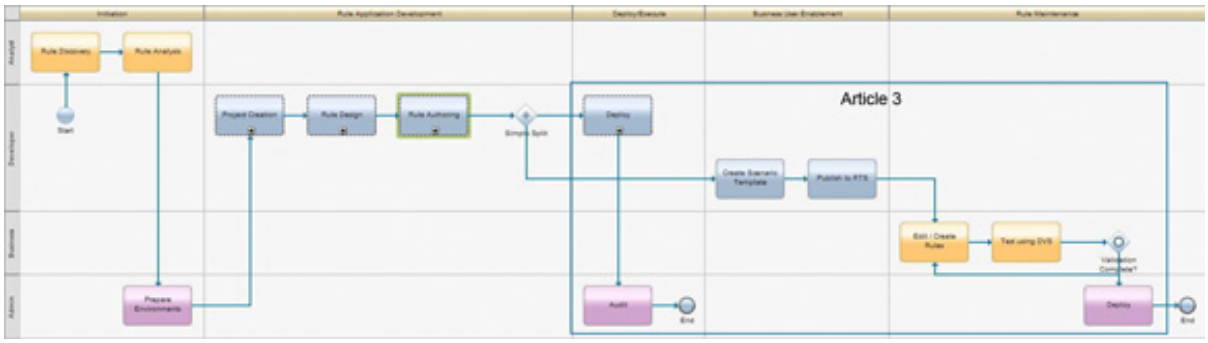
Part 1 of this series introduced you to the smarter city scenario and the use case requirements. We described the role of the decision subsystem and the rationale for selecting IBM WebSphere® ILOG® JRules as a business rule management system (BRMS) to implement the decision subsystem. Part 2 covered the process of rules development. Now, learn how to deploy the initial rule artifacts and enable non-technical business users to continue rule development and testing.

Introduction

In this article, the final in this series, we detail the process for the technical developer to deploy the initial rule artifacts and enable non-technical business users to continue rule development and testing.

Figure 1 shows the portion of the overall decision service development process covered in this article.

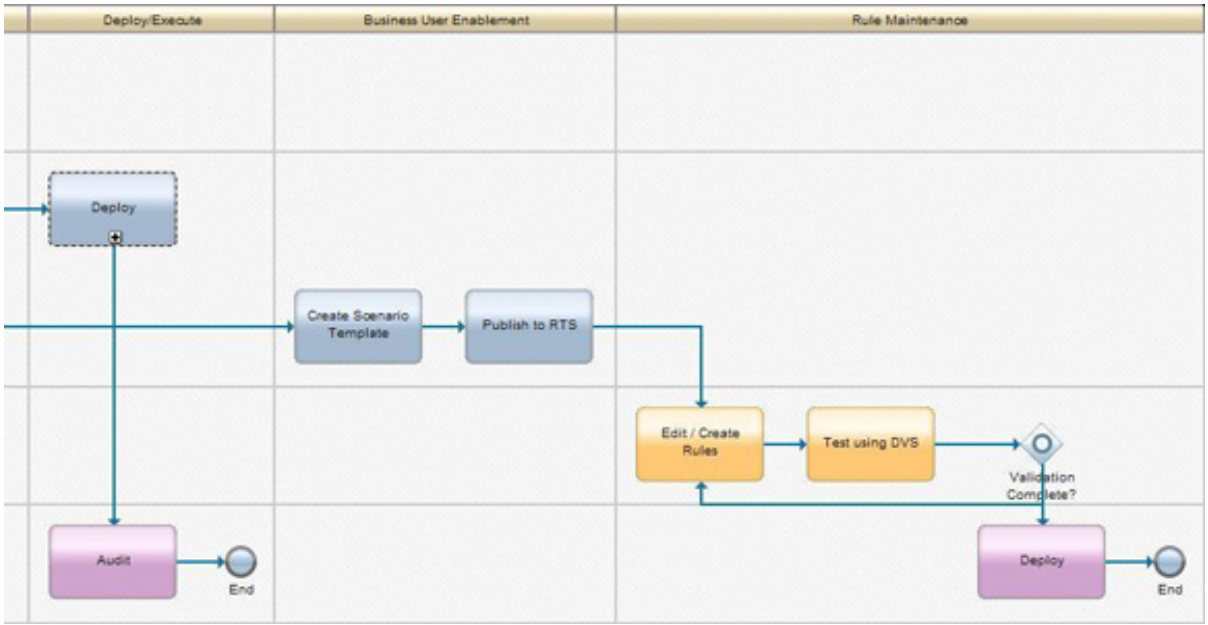
Figure 1. Processes covered in this article



(View a larger version of Figure 1.)

Figure 2 shows a close-up of the rule development tasks covered in this article.

Figure 2. Zoomed-in view of processes covered in this article



(View a larger version of Figure 2.)

Let's jump right in where we left off in the second article.

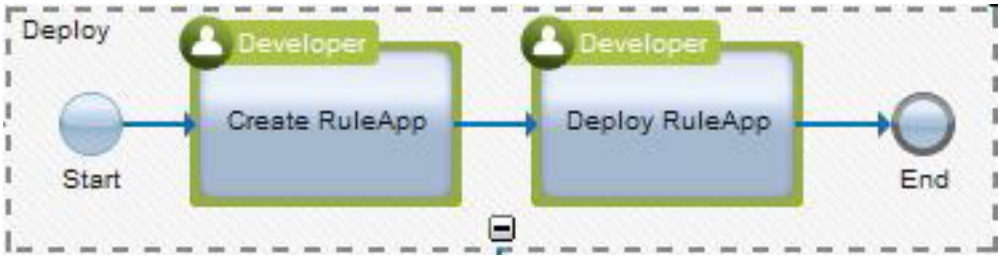
RuleApp

Rulesets are deployed to the Rule Execution Server to enable external distributed clients to invoke the decision service. A RuleApp is the deployable management unit that contains one or more rule sets. From a physical point of view, both rule sets and RuleApps are JAR files containing rule artifacts. At this stage of the process, a rule developer creates a RuleApp project to generate a RuleApp and deploys it to the development environment. WebSphere ILOG JRules includes tools to deploy rules on Java™ SE or Java EE platforms. For this case study, we deploy to the

WebSphere Application Server Community Edition that is distributed with WebSphere ILOG JRules. As illustrated in Figure 3, the deployment process consists of:

1. Creating a RuleApp project
2. Deploying a RuleApp to the Rule Execution Server

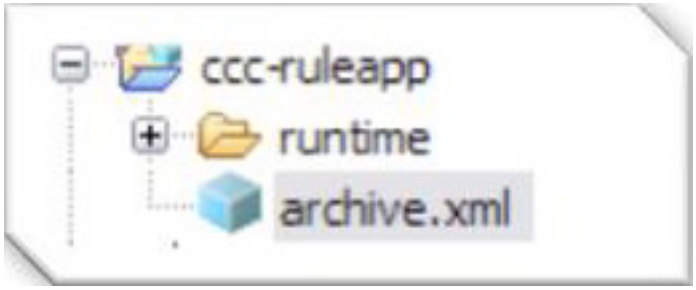
Figure 3. Rule deployment tasks



Create RuleApp project

A RuleApp project called "ccc-ruleapp" is created using a wizard invoked by selecting **New** - **Other** - **RuleApp Project**. In the wizard, "ccc-rules" is chosen as the rule project that is to be included in the RuleApp. This creates a RuleApp project as shown in Figure 4.

Figure 4. RuleApp project



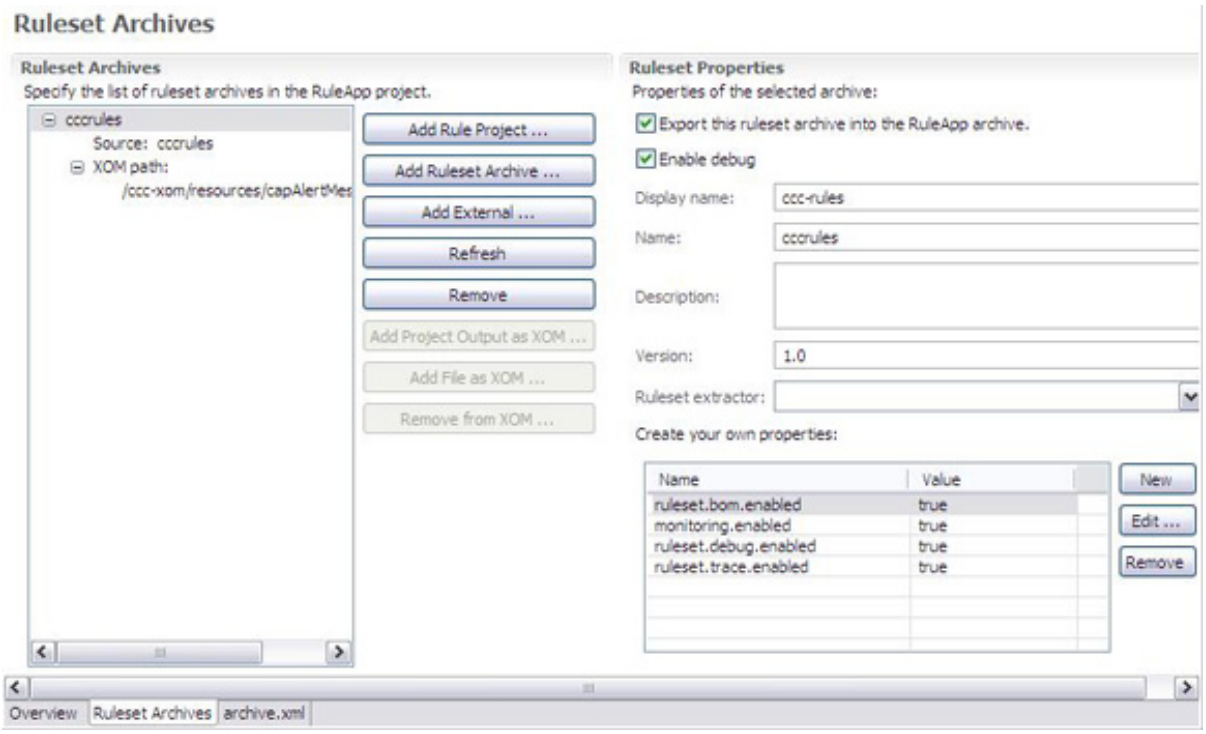
Double-clicking on **archive.xml** opens up the properties of the RuleApp. In the **Ruleset Archives** tab, the ruleset properties can be set to control runtime behavior of the ruleset. To enable monitoring of the ruleset (as described later), the following ruleset properties are added:

```
ruleset.bom.enabled = true

monitoring.enabled = true

ruleset.debug.enabled = true

ruleset.trace.enabled = true
```

Figure 5 displays the ruleset archive properties of ccc-ruleapp after these properties

have been specified.

Figure 5. Ruleset archive properties



(View a larger version of Figure 5.)

Deploy RuleApp

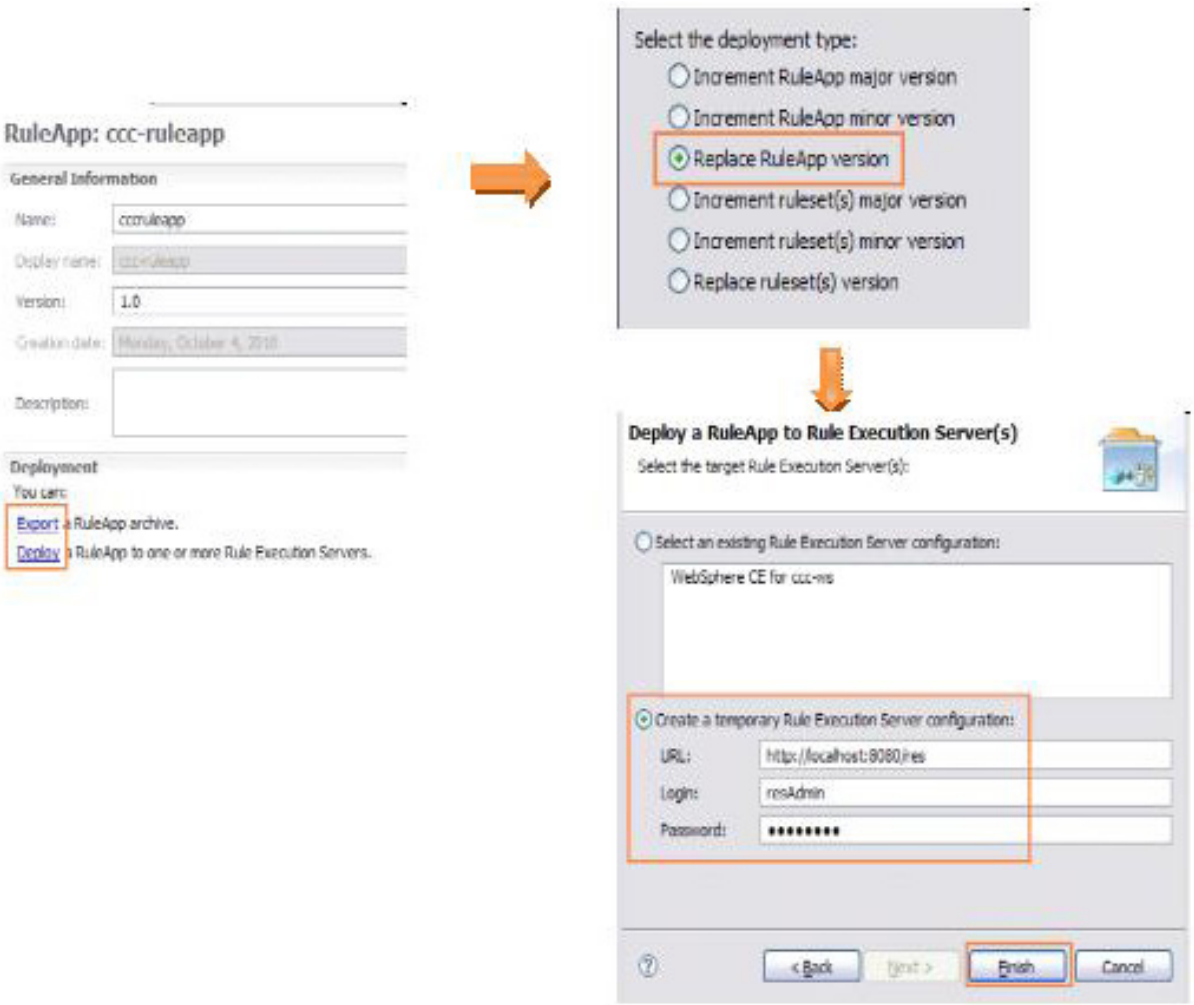
RuleApps can be easily deployed from Rule Studio using the RuleApp editor that is invoked by double-clicking **archive.xml** in the RuleApp project. From the RuleApp editor, click **deploy** to directly deploy the RuleApp to a running application server, which can be remote or local. Of course, this application server should have Rule Execution Server installed and configured on it, and you need to know the authentication credentials. For the case study, we use the WebSphere Application Server Community Edition that is bundled with WebSphere ILOG JRules and already has the Rule Execution Server installed on it, and we use the default credentials for authentication in the Rule Execution Server.

Figure 6 illustrates the sequence of steps used in deploying a RuleApp to the Rule Execution Server. To deploy the RuleApp, follow these steps:

1. Select the **Deploy** hyperlink from the **Deployment** section in the overview of ccc-ruleapp.
2. In the window that opens, select **Replace RuleApp version** as the deployment type and click Next.

3. In the next window, select the **Create a temporary Rule Execution Server configuration** radio button, provide and accept the default credentials (“resAdmin” for Login and Password), and click Finish.

Figure 6. RuleApp deployment

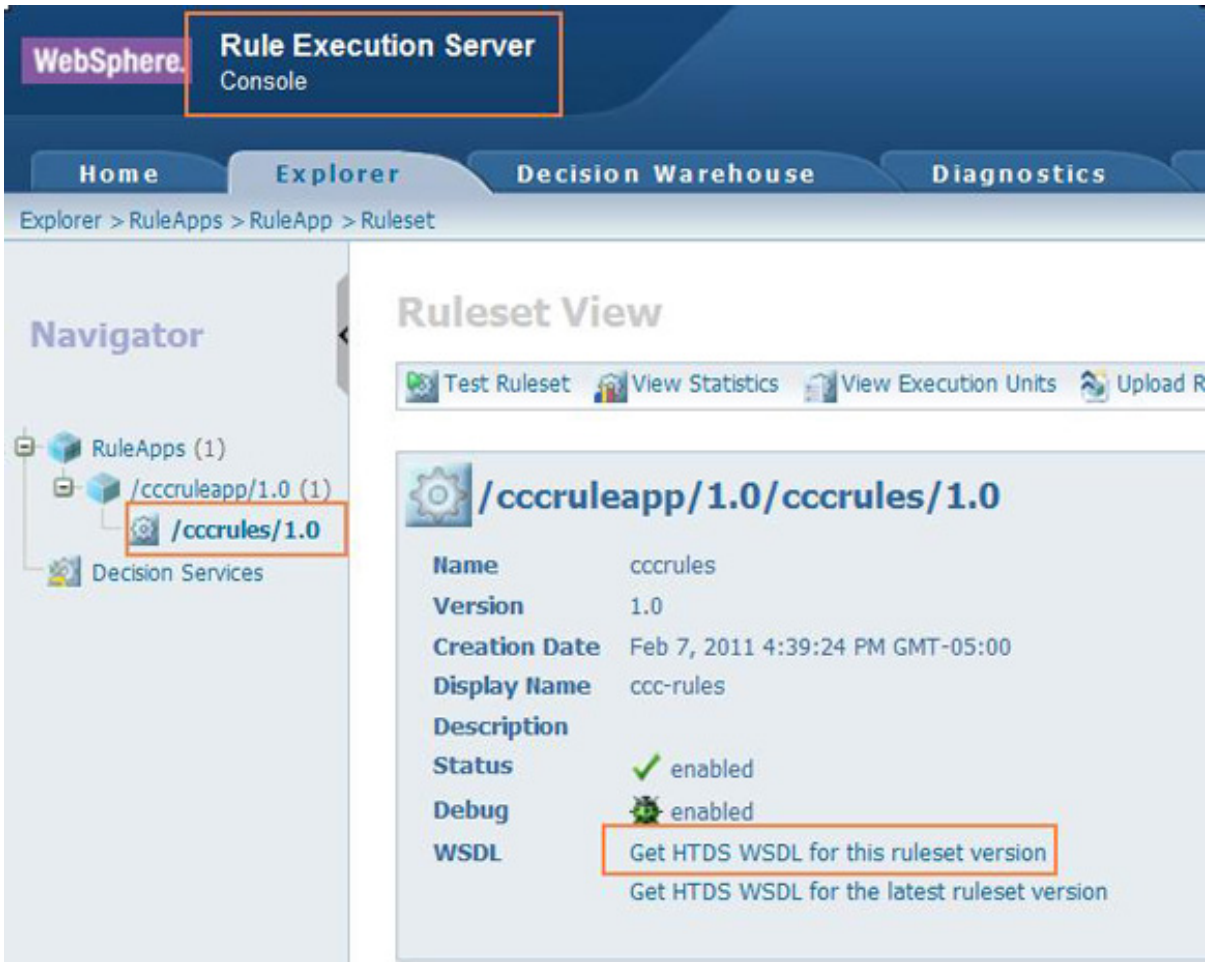


(View a larger version of Figure 6.)

Invoking the RuleApp using the Rule Execution Server

Now, you can manage and monitor the RuleApp using the Rule Execution Server console, which is accessed at http.//<res_server>.8080/res. From the console, we can explore the RuleApps deployed on the server. In our case study, because we have a dynamic XOM, that is, one based on an XSD, the Rule Execution Server automatically exposes it as a web service for invocation through external clients. The WSDL interface to the web service can be obtained through the Rule Execution Server console by clicking **Get HTDS WSDL for this ruleset version** in the **Ruleset View**, as shown in Figure 7.

Figure 7. Rule Execution Server console



At this point, the WSDL is distributed to the external clients and the decision service is ready to be invoked by these external clients. For some quick and simple testing, we can use tools like SOAP UI to issue XML requests to this decision service. Listing 1 provides a sample SOAP request that can be used with SOAP UI.

Listing 1. SOAP request sample

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
          xmlns:dec="http://www.ilog.com/rules/DecisionService"
          xmlns:par="http://www.ilog.com/rules/param"
          xmlns:urn="urn:oasis:names:tc:emergency:cap:1.1"
          xmlns="urn:oasis:names:tc:emergency:cap:1.1">
   <soapenv:Header/>
   <soapenv:Body>
      <dec:DecisionServiceRequest>
         <!--Optional:-->
         <dec:DecisionID>5</dec:DecisionID>
         <par:request>
            <alert>
               <identifier>MSG1</identifier>
               <sender>CityCommandCenter-EPS</sender>
               <sent>2010-10-05T02:30:00+00:00</sent>
               <status>System</status>
               <msgType>Alert</msgType>
```

```
                <scope>Private</scope>
                <addresses>CityCommandCenter-BRE</addresses>
                <info>
                   <language>en-US</language>
                   <category>Met</category>
                   <event>HeavyRainfall</event>
                   <responseType>Assess</responseType>
                   <urgency>Immediate</urgency>
                   <severity>Severe</severity>
                   <certainty>Observed</certainty>
                   <expires>2010-11-15T16:00:00+00:00</expires>
                   <senderName>NATIONAL WEATHER SERVICE ROTTERDAM NL</senderName>
                   <headline>SEVERE THUNDERSTORM WARNING</headline>
                   <description>AT 254 PM PDT...NATIONAL WEATHER SERVICE DOPPLER RADAR
                     INDICATED A SEVERE THUNDERSTORM OVER ROTTERDAM CITY..
                     OR ABOUT 18 MILES SOUTHEAST OF ROTTERDAM...MOVING SOUTHWEST AT 5 MPH.
                     HAIL...INTENSE RAIN AND STRONG DAMAGING WINDS ARE LIKELY WITH
                     THIS STORM.</description>
                   <instruction>TAKE COVER IN A SUBSTANTIAL SHELTER UNTIL THE
                     STORM PASSES.</instruction>
                   <contact>CITY/WEATHERFCT</contact>
                   <parameter>
                      <valueName>RainfallLevel1H</valueName>
                      <value>10</value>
                   </parameter>
                   <parameter>
                      <valueName>RainfallLevel6H</valueName>
                      <value>60</value>
                   </parameter>
                   <parameter>
                      <valueName>RainfallLevel12H</valueName>
                      <value>120</value>
                   </parameter>
                   <resource>
                      <resourceDesc>888001</resourceDesc>
                   </resource>
                   <area>
                      <areaDesc>EXTREME NORTH CENTRAL ROTTERDAM CITY IN NL</areaDesc>
                      <polygon>51.966667,4.333333 51.833333,4.333333 51.833333,4.583333
                         51.966667,4.583333 51.966667,4.333333</polygon>
                   </area>
                   <area>
                      <areaDesc>999001</areaDesc>
                      <polygon>51.966667,4.333333 51.966667,4.458333 51.900000,4.458333
                         51.900000,4.333333 51.966667,4.333333</polygon>
                   </area>
                   <area>
                      <areaDesc>999002</areaDesc>
                      <polygon>51.966667,4.333333 51.966667,4.458333 51.900000,4.458333
                         51.900000,4.333333 51.966667,4.333333</polygon>
                   </area>
                </info>
             </alert>
          </par:request>
       </dec:DecisionServiceRequest>
    </soapenv:Body>
 </soapenv:Envelope>
```

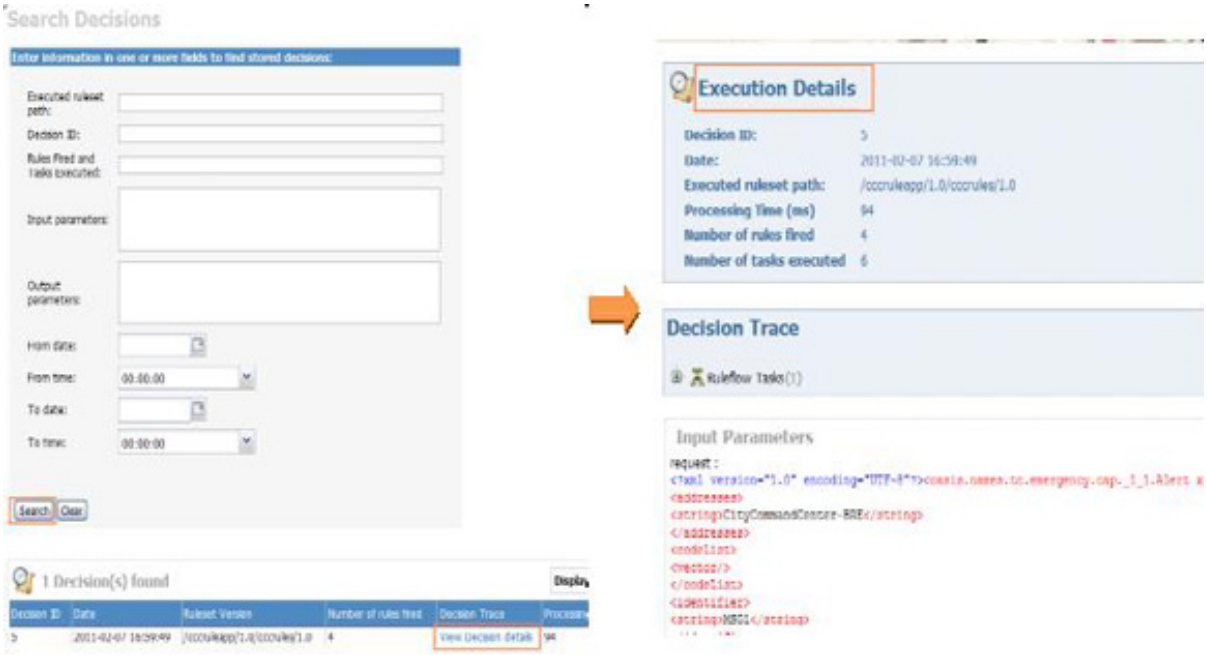
Audit

Administrators can manage the Rule Execution Server using the console. In addition, using the **Decision Warehouse** tab in the Rule Execution Server console, you can audit rule executions for rule sets that have monitoring enabled (see "Create RuleApp project'). You can search rule executions based on date, input/output content, or rules that were fired. You can view details of the rule execution, including

the complete request, response, and all the rules that were invoked. This is very useful in retracing why a decision was made in the past for a particular request, for example, to determine why notifications were sent to the Sewer Department four times during the last week.

Figure 8 depicts the steps needed to view the execution details. From the **Decision Warehouse** tab, specify the filtering criteria and click **Search**. From the search results that are returned, select the one that you are interested in and click **View Decision details**. This action displays a window that has all the rule execution details for the selected item.

Figure 8. The Decision Warehouse tab



(View a larger version of Figure 8.)

Business user enablement

One of the key benefits of a BRMS is that nontechnical business users can maintain the rules independently of the technical team. Business users can use Rule Team Server for this. To enable business users to write and test rules, a rule developer publishes the rule project to the Rule Team Server and creates a Microsoft® Excel® scenario template for business users to build test scenarios.

Create scenario template

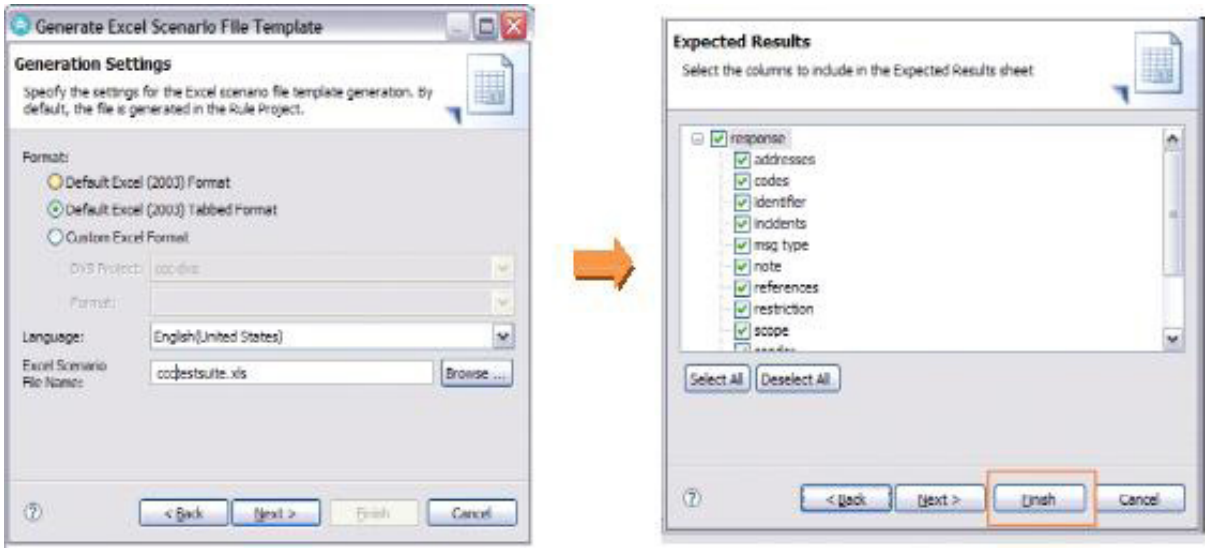
The Excel scenario template for creating Decision Validation Services (DVS) test cases is easily created using a wizard by right-clicking the rule project and selecting **Decision Validation Services - Generate Excel Scenario File Template**. Figure 9

illustrates the options used in each step of the sequence.

1. In **Generation Settings**, choose **Default Excel (2003) Tabbed Format**, specify a file name of your choice in **Excel Scenario File Name**, and click **Next**.

2. In the following screen where you select the columns to include in the expected results worksheet, click **Select All**and click **Finish**.

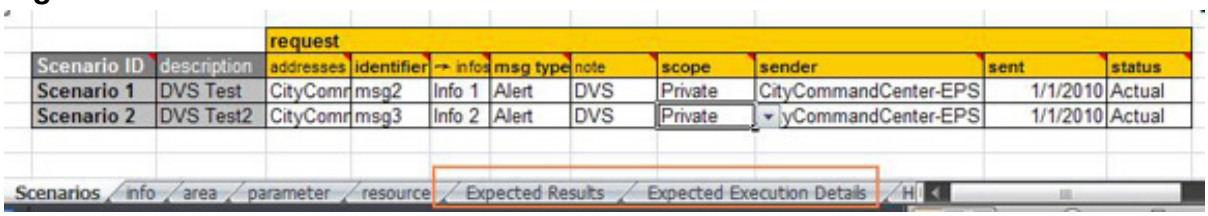
Figure 9. Generate Excel scenario template



(View a larger version of Figure 9.)

This process generates an Excel template with several worksheets, each worksheet corresponding to an object used in the model, such as `info` and `area`. Using this template, scenarios can be defined along with expected results. A sample set of scenarios can be found in the downloadable workspace (see Download). Figure 10 displays the **Scenarios** tab of this sample spreadsheet, which defines two scenarios.

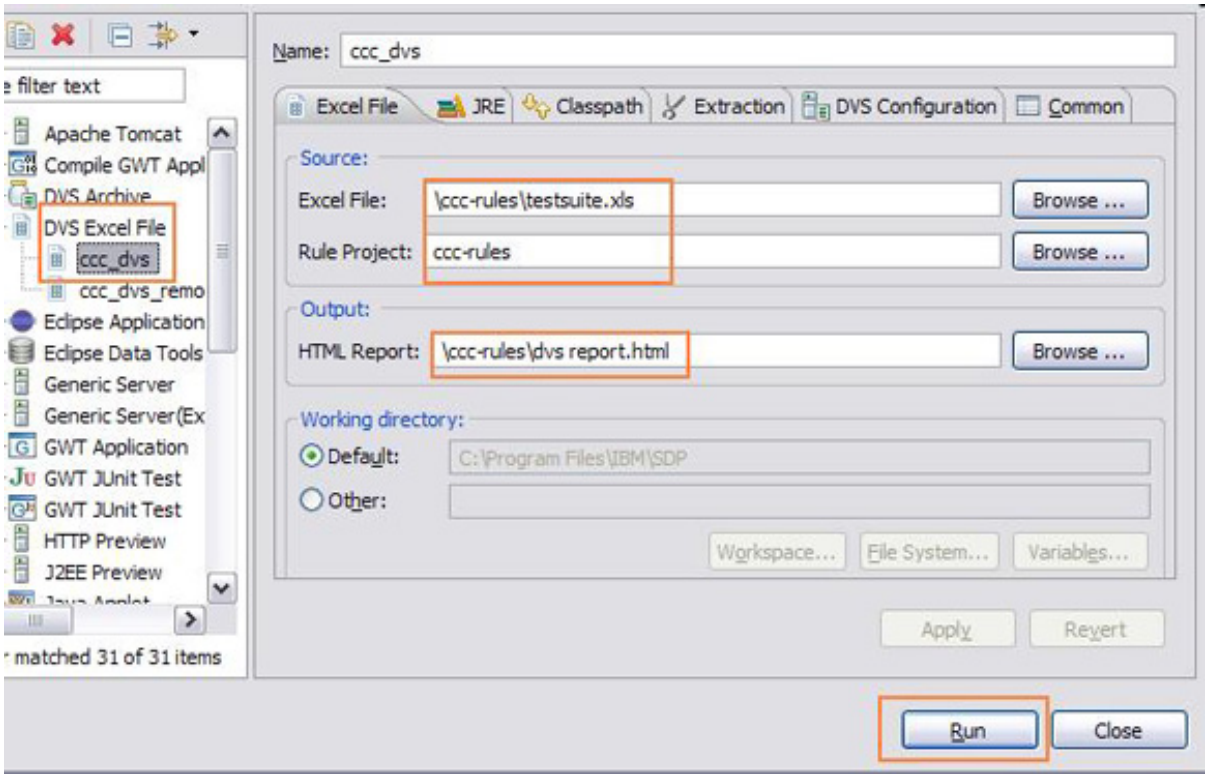
Figure 10.Test scenarios



These scenarios can be run from either the Rule Studio or the Rule Team Server. To run from the Rule Studio, create a **Run Configuration under DVS Excel File** as shown in Figure 11. Use this sequence of steps to create a run configuration:

1. From the main menu on Rule Studio, select **Run – Run Configurations**.
2. elect **DVS Excel File** on the left navigator panel and click the **Launch New Configuration** icon on top.
3. Specify names of your choice for **Name**, **Excel File**, and **HTML Report**. Select ccc-rules for **Rule Project**. Click **Apply**, and then click **Run**.

Figure 11. DVS run configuration in Rule Studio



After you run this configuration, a report, shown in Figure 12, is created in the location specified in the run configuration that contains information about the test runs.

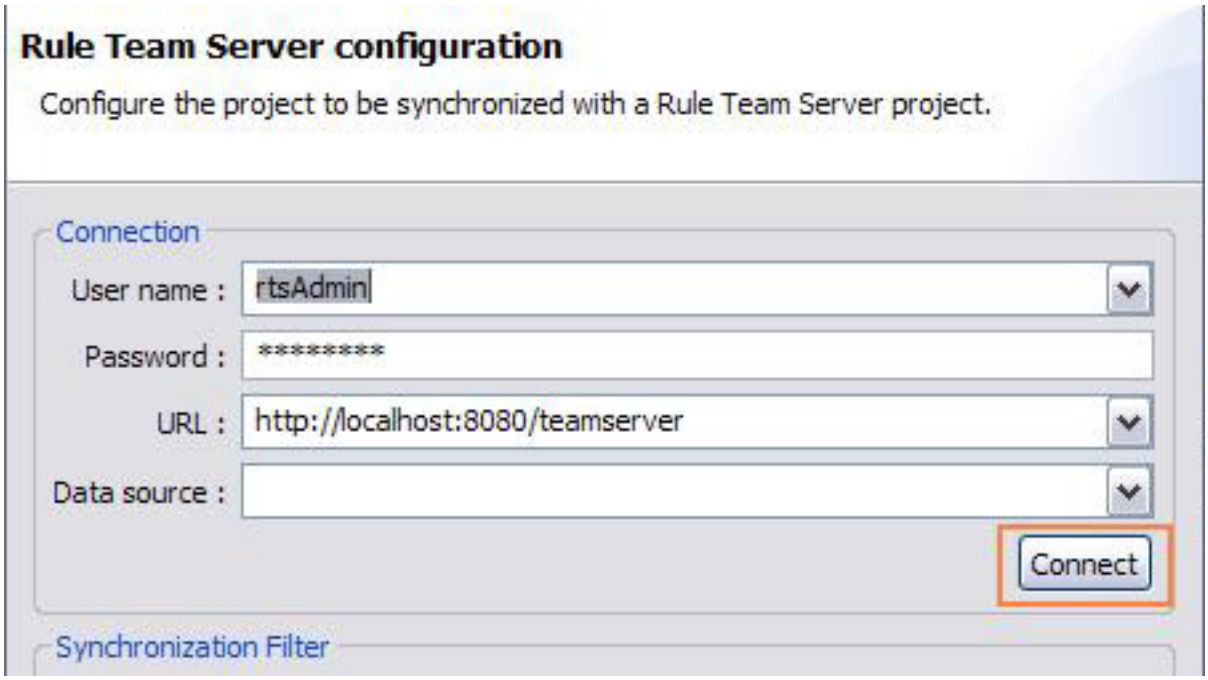
Figure 12. DVS test report

Publish to Rule Team Server

Business users use the Rule Team Server to maintain rules. Developers enable this ability by publishing the rule projects from the Rule Studio to the Rule Team Server repository, which is a database connected to the Rule Team Server.

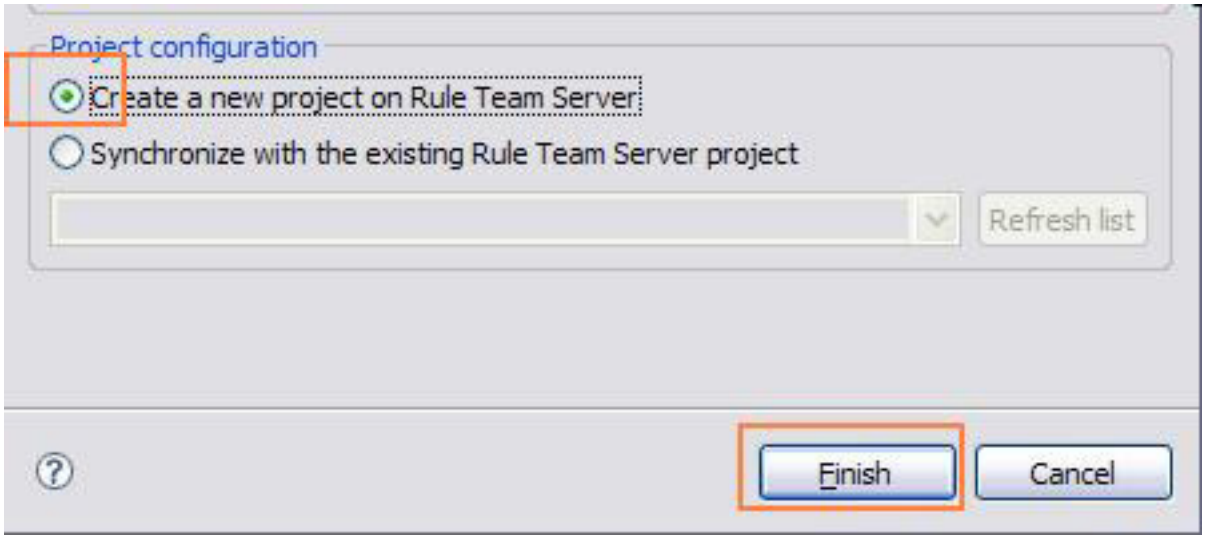
First, we connect to the Rule Team Server from the Rule Studio by right-clicking the rule project and selecting **Rule Team Server - Connect**. This selection opens a window in which you enter the Rule Team Server location and authentication credentials. This process is shown in Figure 13, where we use the default user name and password ("resAdmin"), specify http://localhost:8080/teamserver as the URL, and then click **Connect**.

Figure 13. Connect to Rule Team Server from Rule Studio



After the connection is established, the same window provides options to publishing the rule project on Rule Team Server. As shown in Figure 14, select the **Create a new project on Rule Team Server** option in the project configuration section of the window and click **Finish**.

Figure 14. Publish rule project to Rule Team Server



Synchronization of a rule project can be initiated at any point. Future synchronizations use the **Synchronize with existing Rule Team Server project** option.

For our case study, we have two rule projects: ccc-bom and ccc-rules. Both of these rule projects should be created in Rule Team Server in the manner described above,

even though only ccc-rules is actually modified by business users.

Rule maintenance

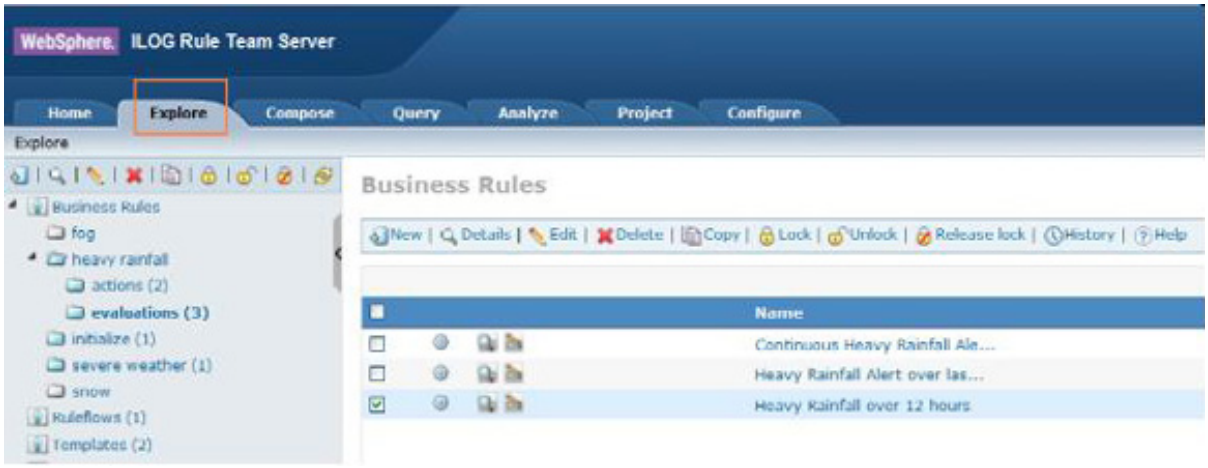
Rule Team Server is a scalable rule management server and repository with a collaborative web environment for authoring, managing, validating, and deploying business rules. It provides a central storage system for business rules and their metadata, and is the designated workspace where business users can collaborate to write, edit, organize, and search for business rules. Business users with QA responsibilities can create test scenarios and launch them in DVS from the Rule Team Server. Administrators use the Rule Team Server to extract rules from the repository and deploy them to various environments.

Maintain rules

Policy managers use Rule Team Server to work collaboratively on rule artifacts. Rule Team Server provides history and versioning services that support auditing and rollback of rule artifacts.

Select the **Explorer** tab in the Rule Team Server to browse rules and edit them. As seen in Figure 15, the rule packages are arranged in a hierarchical manner in the left navigation panel and the rules in the selected rule package are shown in the right panel.

Figure 15. Browse rules in Rule Team Server



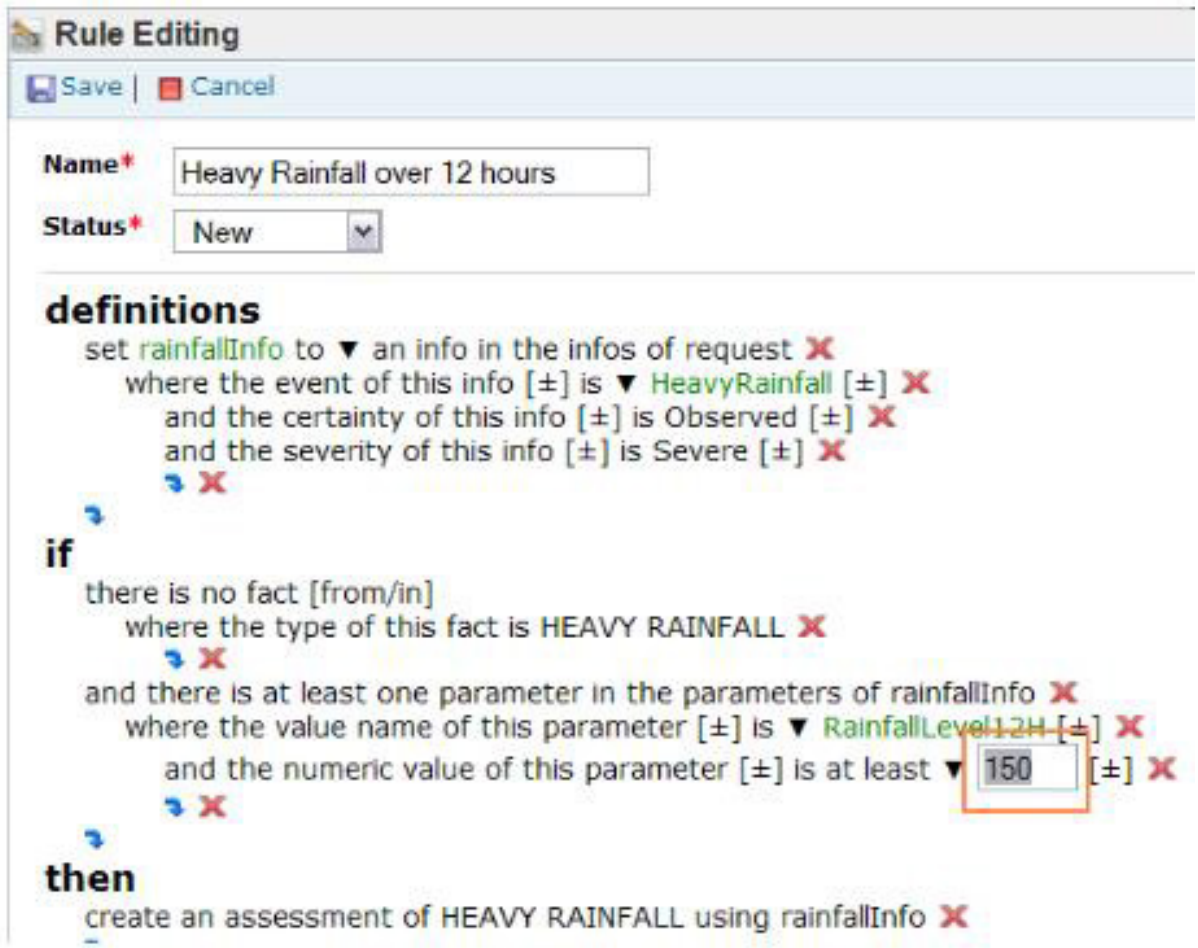
(View a larger version of Figure 15.)

Rules can be directly edited in this tool. For example, if the rule that assesses heavy rainfall based on rainfall over the last 12 hours changes from 150 mm to 160 mm, a business user can directly make this change to the rule *Heavy Rainfall over 12 hours* in the Rule Team Server.

Figure 16 shows the rule being modified in the Rule Team Server. The steps involved are these:

1. Select the rule and click **Edit**. Alternatively, click the edit icon next to the rule.
2. In the Rule Editing window, click **150** and change it to "160".
3. Click **Save**.

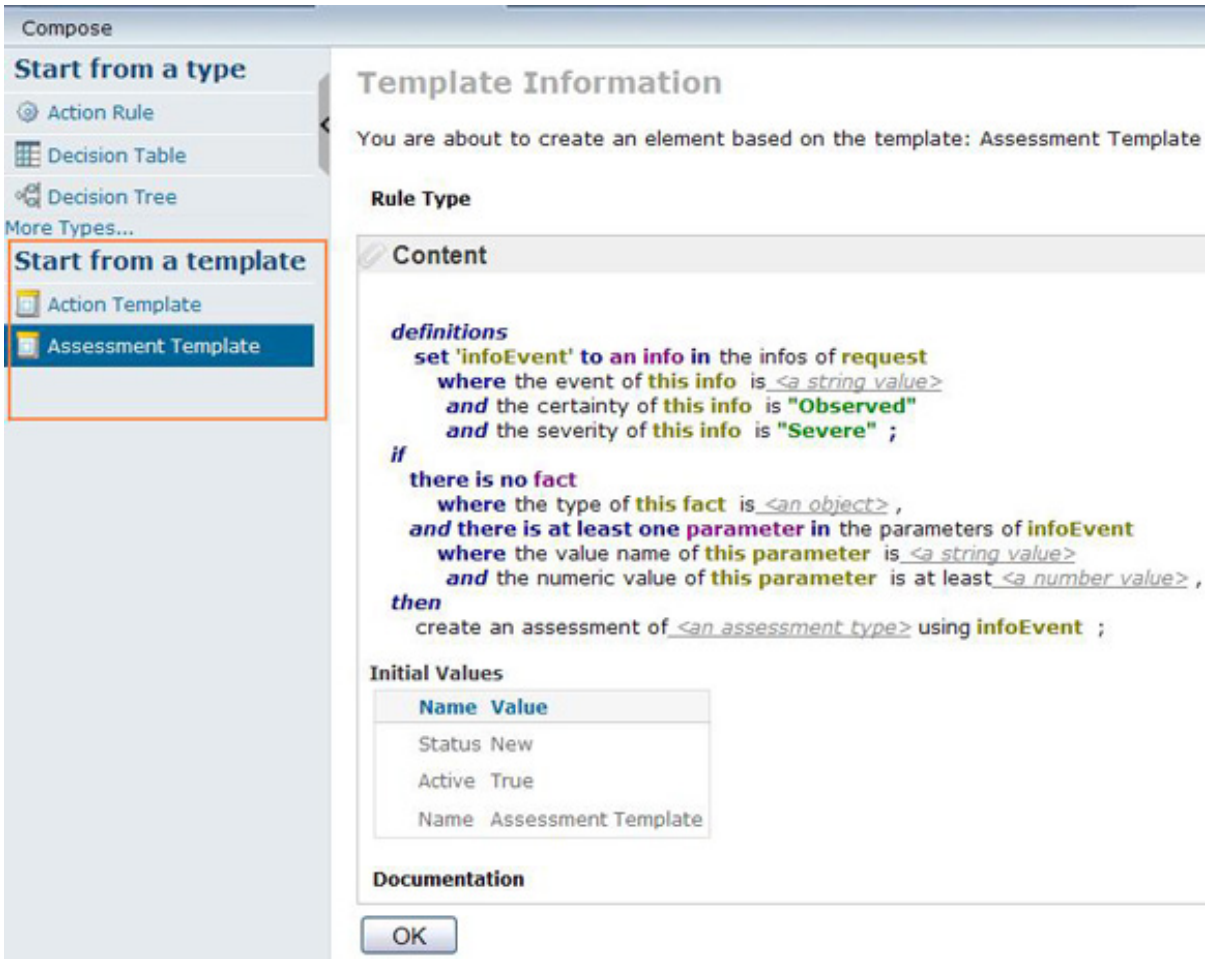
Figure 16. Editing a rule in the Rule Team Server



(View a larger version of Figure 16.)

In addition, business users can create new rules from scratch, or more easily, using a predefined rule template. For example, Figure 17 shows a new assessment rule being created from a predefined rule template called *Assessment Template*. Various parts of the rule are already created and frozen in the rule template. To complete defining the rule, a business user only needs to fill out the placeholders.

Figure 17. Create new rule from a template

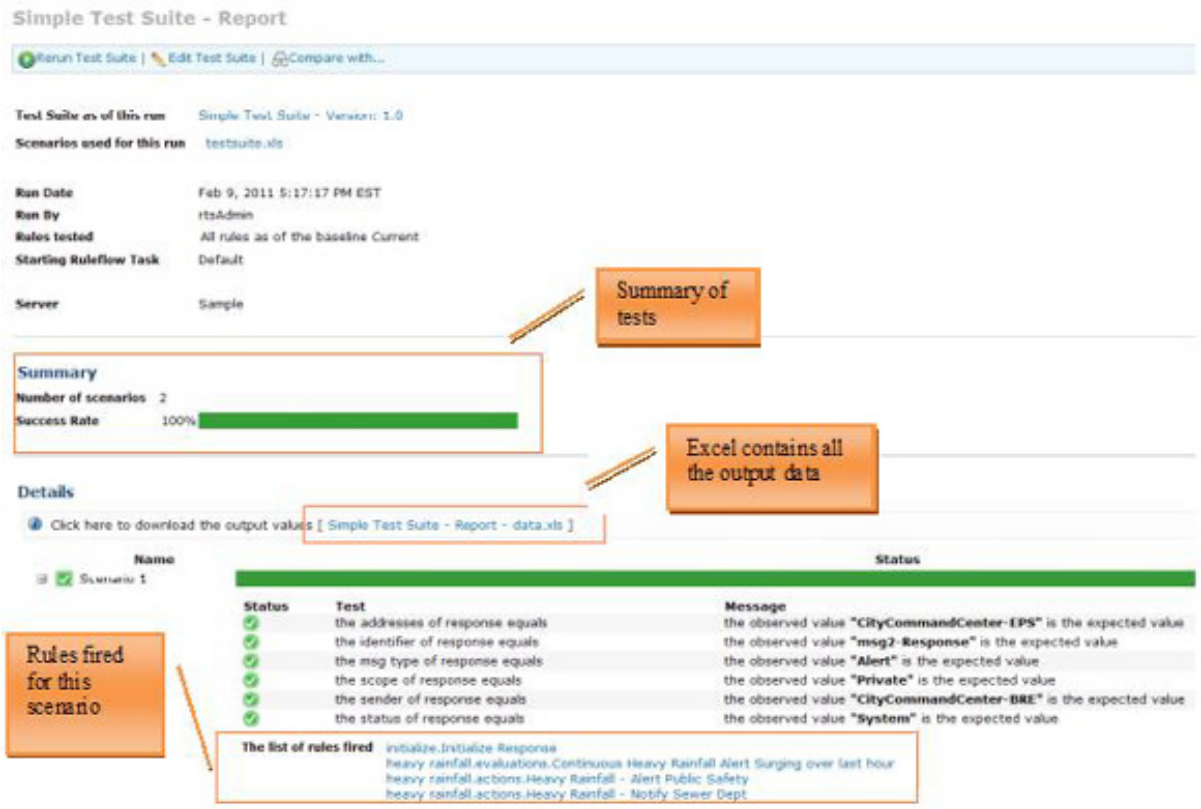


Test

Decision Validation Services is used for creating test scenarios for developers, QA teams, and business users to validate correctness and effectiveness of rulesets. Using the Excel template provided by rule developers, business users create unit test cases as rows in Excel spreadsheets. Decision Validation Services work in the Rule Team Server by connecting to a server that has Rule Execution Server and a Scenario Service Provider (SSP).

To start, create a new test suite in the Rule Team Server, which essentially consists of pointing to the Excel spreadsheet that contains the test scenarios. When this test suite is run, it extracts rules and temporarily deploys them to the specified Rule Execution Server, which is typically in the QA environment. Then each of the scenarios defined in the spreadsheet is run as an individual test case against this freshly deployed ruleset, and the results are displayed in a report. A sample report is shown in Figure 18.

Figure 18. DVS report



(View a larger version of Figure 18.)

Deploy

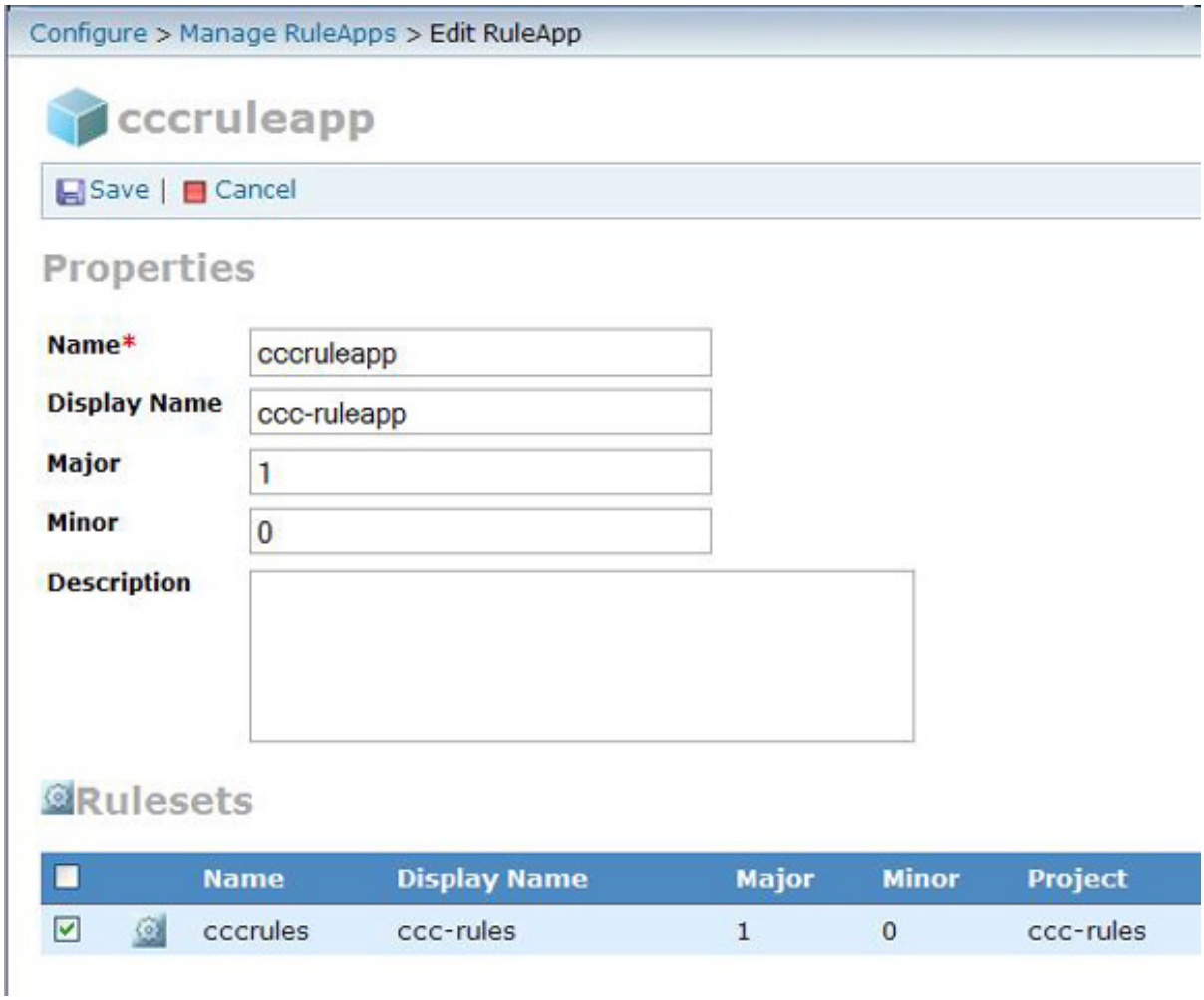
After rules have been validated and are ready for deployment, you can deploy RuleApps directly from the Rule Team Server; of course, this approach assumes that you have the appropriate credentials for doing so.

Deployment follows two main scenarios:

- Hot deployment, when you want an immediate availability of rules.
- Staged deployment, when you want to deploy to a controlled production environment.

After selecting the **Configure** tab in Rule Team Server, an administrator can create or edit a RuleApp. Figure 19 shows a RuleApp where the name of the RuleApp is set to cccruleapp and cccrules is defined as the contained ruleset.

Figure 19. Create a RuleApp in Rule Team Server



A RuleApp can now be deployed to any host running the Rule Execution Server or simply exported to a RuleApp JAR file for staged deployment. Figure 20 shows hot deployment to the local Rule Execution Server.

Figure 20. Deploy from Rule Team Server

After a RuleApp is deployed, it is available to external clients on that environment as a decision service to make complex decisions. Note that with hot deployment, subsequent invocations from external clients use the new rule set for processing.

Conclusion

We have seen that WebSphere ILOG JRules is a powerful tool that offers a rich set of features for building business rules management systems that can be executed in an SOA environment. Using a case study, we walked through a commonly used decision service development process to gain an appreciation of how people in different roles work together and how WebSphere ILOG JRules, through its modules, wizards, and accelerators, eases many of the tasks.

Downloads

Description	Name	Size	Download method
	cccrules_pif_051811.zip		HTTP

Information about download methods

Resources

Learn

- IBM WebSphere ILOG JRules product page: Access more about the features and benefits, system requirements, and support from the product home page.
- IBM WebSphere ILOG JRules Information Center: Find more information about this product line and its features.
- OASIS Common Alerting Protocol 1.1: The Common Alerting Protocol (CAP) is a simple but general format for exchanging all-hazard emergency alerts and public warnings over all kinds of networks. See the October 2007 update too.
- Policies and Rules – Improving business agility: Part 1: Support for business agility (Hondo, Boyer, Ritchie, developerWorks, March 2010): One challenge in architecting and implementing agility in business solutions today is that the use of the terms, policy, and rule, differs across products. Learn the concepts and relationships of policies and rules technologies to implement specific business strategies and tactics.
- Creating intermediate facts in WebSphere ILOG JRules using synthetic objects (Raj Rao, developerWorks, November 2010): See this WebSphere Technical Journal article for more information about virtual classes to implement synthetic objects.
- IBM developerWorks Industries: Get all the latest industry-specific technical resources for developers.
- developerWorks technical events and webcasts: Stay current with technology in these sessions.
- developerWorks on Twitter: Join today to follow developerWorks tweets.
- developerWorks podcasts: Listen to interesting interviews and discussions for software developers.

Get products and technologies

- WebSphere ILOG JRules V7.1: Get the trial download.
- IBM product evaluation versions: Download or explore the online trials in the IBM SOA Sandbox, and get your hands on application development tools and middleware products from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.

Discuss

- developerWorks blogs: Check out these blogs and get involved.

About the authors

Raj Rao

Rajesh (Raj) Rao has been working in the area of expert systems and business rule management systems for over 20 years, during which time he has applied business rules technology to build diagnostic, scheduling, qualification, and configuration applications across various domains such as manufacturing, transportation, and finance. He has been with IBM since 2009. With a background in artificial intelligence, his interests include natural language processing and semantic web.

Sandeep Desai

Sandeep Desai is a Senior Certified Enterprise IT Architect with IBM WebSphere's Business Partner Technical Professional team. He works with strategic business partners from startup to large firms. He mentors them to SOA-enable their solution. He evangelizes, educates, and enables partners on IBM software platform. Sandeep is Open Group Distinguished Certified IT Architect, IBM Senior Certified Enterprise IT Architect, and IBM Certified SOA Solution Designer.