Capturing Rule Execution Records for Auditing and Analysis

Progress Corticon enables you to store data about a Decision Service's execution, input and output payloads, as well as rule messages that are generated during execution. This data can later be used for auditing and analysis.

Video: http://documentation.progress.com/output/video/Corticon/Cort_CapturingRuleExecutionRecords.html

Transcript:

Corticon enables you to store rule execution data for auditing and analysis. Imagine a Decision Service that processes customer orders at an online retail store. Each time a customer places an order, the Decision Service performs a number of operations. It checks if the customer is a preferred card holder. It processes coupons offering $10 off on the next order if the customer's shopping cart is worth $75 or more. And so on. Storing rule execution data in a database can be valuable. It enables you to analyze and gain important insight into the behavior of your rules. For instance, you could use it to trace the execution sequence that led to a particular result.

In this video, you will see how to set up Corticon 5.5.2 to store a Decision Service's execution data to a database.

To record Decision Service executions, we need to perform three steps:

- First, we need to set up the database schema.
- Then, we need to enable Corticon Server's **Execution Recording Service**.
- Finally, we need to configure the Decision Service's properties to enable the recording service to write its execution data to the database.

Corticon Server can write to any of the supported databases, such as Progress OpenEdge, Microsoft SQL Server, Oracle, and IBM DB2. Here, we will use a Microsoft SQL Server 2014 Express database named **CorticonExecutionRecords** that we have already created.

To begin, we create a schema for the database using Corticon Studio.

We select **Project > Create Execution Recording Schema**. This opens the **Create Schema** dialog box. Here, we specify the **Database Server**, which in our case is **SQL Server 2014**. We define the **Database URL**, entering the hostname, which is localhost, and the database name, which is **CorticonExecutionRecords**.

Next, we define the **Username** and **Password**.

Let's click **Test Connection**. This helps us verify that our database connection values are correct.

At this point, we are creating the database schema. Later, when we enable Corticon Server to start the **Execution Recording Service**, we will need to define these same property values in Corticon Server's **brms.properties** file. One way to prepare for that step is to copy these property values into a text editor, so we can use them later. To do this, we click **Show Encrypted Values**, and copy the properties to a text editor. The username and password values are encrypted but that does not matter—Corticon Server will be able to decrypt these values when it reads its brms.properties file.

Finally, we click **Finish**. It takes a few seconds to create the database schema. Once it is ready, we see a message—like this one—stating that the execution recording schema was created successfully.

Let’s take a quick look at the schema. In SQL Server Management Studio, we expand the **CorticonExecutionRecords** database. As you can see, new tables have been added.

The CC_EXECUTION table stores date and time information each time the Decision Service executes. The CC_PAYLOADS table stores the input and output payloads. The CC_RULEMESSAGES table stores rule messages that are generated during execution.

Now that the database schema is ready, let’s move on to Step Two—enabling Corticon Server’s **Execution Recording Service**. To do this, we must modify Corticon Server’s brms.properties file, which is located in Corticon Server’s work directory. We open the file in a text editor and scroll down to the execution recording property.

We uncomment this line and change the property value to **true**. By default, this will enable the Execution Recording Service to record the execution timestamp as well as the payload and rule messages. We can also choose to suppress recording payload by changing the payload recording property to **false** and suppress recording rule messages by changing the rule messages property to false. Because we want to record everything, we do not modify these properties.

Next, we need to define the database connection properties. Recall that earlier we copied these values from Corticon Studio into a text editor. We’ll now insert them at the end of Corticon Server’s brms.properties file.

Let’s save the file.

Corticon Server’s Execution Recording Service is enabled.

Now, we come to our final step—configuring the Decision Service’s properties to enable recording. There are a number of ways to set this property. In this video, we’ll configure the property while adding the Decision Service to a Web Console application.

To prepare for deploying the Decision Service through the Web Console, we first need to package our Ruleflow into an **EDS** file using Corticon Studio. An EDS file is a precompiled package that contains all assets that Corticon Server needs to deploy and run a Decision Service.

Here is our Ruleflow, called **OrderRules**, that we need to package. We select **Project > Package and Deploy Decision Services**. This opens the **Package and Deploy Decision Services** wizard, where we select **Package and save for later deployment** and click **Next**. We then select our Ruleflow, specify a path to store the EDS file, and click **Next**. Finally, we click **Finish**.

The Ruleflow is now packaged into a deployable EDS file, as you can see here.

Next, we deploy the EDS file using the Web Console. We have already started Corticon Server. So we open the Web Console and create a new application in the **Applications** tab. Let's call the application **MyApplication**.

Since our Corticon Server is installed on the same computer as the Web Console, we select **local server** in the **Servers** drop-down.

Now, let's add the Decision Service to the application. We click **Add** in the **Decision Services** section. This opens the **Add Decision Service** pop-up, where we:

- Give a name to our Decision Service,
- Choose the EDS file that we created earlier,
- And set the maximum pool size for the Decision Service.

Right after the maximum pool size property is a property called **Use Execution Recording Service**. This is where we set the property to enable execution recording. We check this property, retain the default settings for the other properties, and click **Add**.

This brings us back to the **Application** page, where we now see a new button called **Save & Deploy**. We click this button to deploy the application containing our Decision Service to Corticon Server.

Now, all three steps are done. We have created the database schema. We have set up Corticon Server to start its execution recording service. And, we have configured our Decision Service's properties to enable the recording service to write its execution data to the database.

Let's test it by invoking this Decision Service. In the Web Console, we select our Decision Service, click **Test Execution**, and specify a SOAP request file that we created earlier for testing.

As you can see, the XML message gets displayed in the **Request** area. We delete the SOAP tags from the request because the Web Console will add them when it invokes the Decision Service. Finally, we click **Execute**. As you can see, the Decision Service processed the data and sent a response message.

Now, let's verify that the execution data is written to the database. In SQL Server Management Studio, we expand the **CorticonExecutionRecords** database.

We right-click CC_EXECUTION and choose **Select Top 1000 Rows**. As you can see the Decision Service's execution is recorded here with a **Start** and **Stop** timestamp.

Let's run the same query on CC_PAYLOADS. As you can see, the input and output payloads have also been recorded. The **execution ID** column links each payload record to the execution record in the CC_EXECUTION table.

Finally, let's run the same query on CC_RULEMESSAGES. As you can see, the rule messages have also been recorded.

You have now seen how you can capture rule execution data in a database for later auditing and analysis.