TIBCO BusinessEvents® Enterprise Edition

Getting Started

Version 6.1.2
January 2022

Contents

Tutorial Projects

Tutorials in this guide are based on one simplified business scenario. The tasks in the tutorials provide step-by-step instructions and explain the main ideas so you gain understanding as well as practical knowledge. References to related information are provided so you can jump to the detailed documentation to learn more.

The Project Design Tutorial shows you how to configure a TIBCO BusinessEvents project, run it at the command line, and test its behavior. This tutorial focuses on ontology and inferencing.

The Cache OM Tutorial shows you how to add caching functionality to the project.

The Backing Store Tutorial shows you how to add a backing store, which allows the object data generated in the inference engine to be persisted on disk, and reused as needed.

Configured Tutorial Projects are Provided

The tutorial projects, all based on the basic `FraudDetection` example, are located in the `BE_HOME`/examples/standard directory.

Additional Example Projects Provide More Learning

You can explore many examples in the `BE_HOME`/examples directory. These examples demonstrate specific techniques that you can apply in your work.

Skills Required

This guide is written for users with little or no familiarity with TIBCO products. Readers should have some familiarity with Java programming and the Eclipse platform.

Eclipse Platform

If you are not familiar with Eclipse platform, you can learn more from the Eclipse documentation. Open TIBCO BusinessEvents Studio and click **Help > Help Contents** to view a list of manuals, such as Workbench User Guide.

The Fraud Detection Scenario

The tutorial is built on a simplified fraud detection scenario and decision making flow.

To establish whether fraud is suspected, the runtime engine correlates the frequency of and amount of debits in a rolling time window, and flags accounts that satisfy both of the following criteria:

- The account incurs more than three debit transactions in a two-minute period.

- The sum of the debits that occurred in the two minute period totals more than 80% of the average monthly balance of the account.

For the purpose of the tutorial, messages arrive from an HTTP server, provided by TIBCO BusinessEvents. The event correlation is performed using simple TIBCO BusinessEvents rules. Suspicious accounts are simply set to "Suspended." Actions are printed to the monitor so you can see the project in action.

An additional event and rule not shown allow you to unsuspend an account.

Project Design Tutorial

You will configure, build, deploy, and test a TIBCO BusinessEvents project, with an emphasis on the basic project design.

Deployment activities are limited to the basic actions required to test a design.

The section The Fraud Detection Scenario explains the general tutorial scenario, the fraud detection criteria, and how a customer account can become Suspended. This section explains in more technical terms what happens at runtime given an example debit that triggers the fraud detection rules. (You will learn more details about the terms shown in italics below, as you complete the tutorial steps):

1. A message arriving through a TIBCO BusinessEvents channel is transformed into an event. (At design time you create an event type for this purpose, with the appropriate properties.) The event instance is then *asserted into the Rete network*, an in-memory network of objects based on the Rete algorithm, which enables fast matching of *facts* with *rule dependencies*.

2. The presence of this new event in the Rete network causes the inference engine to check for *rules* that are designed to be triggered when this event is asserted.

3. A rule that is triggered by this event executes. A rule might make changes to concept instances, create an event and send it to a channel (and out of the TIBCO BusinessEvents application to some destination), and so on. The rule then consumes the event unless there is a reason to persist the event.

> ℹ **Note:** It is important to consume events when they are no longer needed so that they do not trigger rules to fire erroneously. On the other hand, it is also important to use a long enough time-to-live (TTL) setting for an event, so that it exists long enough to perform all work needed, for example, to trigger rules that correlate multiple events and take appropriate actions.

Import of Existing Projects into Your Workspace

It is recommended that you complete the tutorials yourself, because "learning by doing" is the most effective way to become proficient.

If completing the tutorials is not possible, however, you can read the tutorials and refer to the fully configured example projects, provided here:

BE_HOME/examples/standard/FraudDetection

BE_HOME/examples/standard/FraudDetectionCache

BE_HOME/examples/standard/FraudDetectionStore

Dependency of Interactive Readme File on a Higher Level Directory

The `readme.html` file uses resources in the *BE_HOME*/examples/_resources directory to enable you to send messages to the deployed example project using an HTTP channel. Therefore you must maintain the relative positions of these two directories.

Dependency between Readme File and Project — Port Number

Each example project uses a different port for the HTTP channel, so you can run more than one example at a time on your machine. The readme file interacts with the channel on the specified port at runtime. FraudDetection uses port `8108`, FraudDetectionCache uses port `8109` and FraudDetectionStore uses `8209`.

If you change the example project's port (in the HTTP Connection resource), make sure you change the port in the `readme.html` too. Open the `readme.html` in an editor and edit this line (showing the FraudDetection project port as an example):

```
SendEventForm.setServer("http://localhost:8108");
```

Copy Example Directory Trees before Importing

Because of this dependency between the `readme.html` file and the `_resources` directory (and also because it is good practice to work in copies of the shipped examples), it is recommended that you make a copy of the directory tree for any example you want to work with.

For example, copy *BE_HOME*/examples/standard/FraudDetection, and paste it as a peer of the provided example, renaming the `FraudDetection` directory, for example, to `FraudDetection2`. As an alternative you can copy the entire *BE_HOME*/examples directories and then work with the copied set of examples.

In either case, you can use the `readme.html` file to test the deployed project.

At deploy time the only files that are used are the configured EAR and CDD files. When you construct a command to start an engine, these files must be available.

Importing an Example Project into Your Workspace

You can work on a sample project in TIBCO BusinessEvents Studio after importing the project into the TIBCO BusinessEvents Studio workspace.

Procedure

1. Start TIBCO BusinessEvents Studio. In Windows, click **Start > All Programs > TIBCO > *YourEnvironment* > TIBCO BusinessEvents 6.0 > Studio**.

2. As needed, create a workspace for your tutorial projects.

   Select **File > Switch Workspace > Other** and select the directory you want to use, or create a new directory.

3. Select **File > Import > TIBCO BusinessEvents > Existing TIBCO BusinessEvents Studio Projects**, then click **Next**.

4. In the **Existing project root directory** field, browse to the project directory and click **Select Folder**.

   For example, to import the FraudDetection project from `BE_HOME`/examples/standard/FraudDetection/FraudDetection, select its parent directory (`BE_HOME`/examples/standard/FraudDetection/FraudDetection).

5. Select or clear the **Copy project into workspace** checkbox to suit your needs.

   If you are working from a copy of the example project, as recommended, you do not have to copy projects to your workspace. It's up to you how you manage your files.

6. Click **Finish**.

   The selected projects appear in Studio Explorer.

Before you Create a Project

Before you can create a specific project, you need to create an empty project using TIBCO BusinessEvents Studio.

Learning Points

TIBCO BusinessEvents Studio is the Eclipse-based user interface for TIBCO BusinessEvents. It enables you to build, test, and debug projects. Use of this industry-standard development framework shortens your learning curve and enables you to take advantage of common tools and facilities.

TIBCO BusinessEvents has these Eclipse perspectives:

- TIBCO BusinessEvents Studio Debug
- TIBCO BusinessEvents Studio Development
- TIBCO BusinessEvents Studio Diagram

TIBCO BusinessEvents switches perspectives transparently, depending on the editor you are working with.

In the lower part of the user interface are various tabs used as needed:

Tab	Description
Console	**Console** The Console view displays and logs the errors resulting from improper execution of the TIBCO BusinessEvents engine.
Problems	**Problems** The Problems view reports all validation errors in the project, including language validation errors, access control errors, and so on. It also displays messages relating to project analyzer. Double-click a problem entry to open the associated editor. See Analyzing and Validating the Project for more details.
Error Log	**Error Log** The Error Log view captures all the warnings and errors logged by Eclipse plug-ins. The log file itself has the extension `.log` file and it is stored in the `.metadata` subdirectory of the workspace.
Properties	**Properties** The Properties view shows metadata about a resource. To view the metadata, click the resource in TIBCO BusinessEvents Studio Explorer. Not all resources use this tab.
Terminal	**Terminal** Using the Terminal view you can perform command-line operation (such as, studio-tools) from within BusinessEvents Studio. The terminal also provides shell access to remote systems from within BusinessEvents Studio.

When you create a new project a set of folders is created. You can use a different folder structure to organize your project components in any way you like. Example projects are kept simple and use provided folder names such as Concepts, Events, Rules. More complex projects might use a different folder hierarchy with names that relate to the purpose or contents of the folders

More Information

Project Development in *TIBCO BusinessEvents Developer's Guide* discusses actions you can take at the project level such as importing projects, validating projects, using project libraries, and so on.

Creating the FraudDetection Project

In this task, you will create a new project, place it in a desired location, and save it.

Procedure

1. Start TIBCO BusinessEvents Studio. In Windows, click **Start > All Programs > TIBCO > > *YourEnvironment* > TIBCO BusinessEvents 6.0 > Studio** .

2. The first time you run TIBCO BusinessEvents Studio, a Welcome screen displays.

   You can access this guide from the Welcome page. Click the **X** next to Welcome to dismiss the screen.

3. As needed, create a workspace for your tutorial projects.

   Select **File > Switch Workspace > Other** and select the directory you want to use, or create new.

4. Right-click in the TIBCO BusinessEvents Studio Explorer view (the panel on the left, where the project folders will display), and select **New > Project**.

   You will see the New Project wizard.

5. From the list, select **TIBCO BusinessEvents > Studio Project** and click **Next**.

6. In the New Studio Project dialog, enter the project name FraudDetection, and click **Finish**.

   If you want to use a non-default location, uncheck the Use default location checkbox and select the location.

7. Save the project according to the resources you choose to save:

- To save all changes to all resources in a project (since last save), click **File > Save All** or click **Ctrl+Shift+S**.

- To save changes in just the currently viewed resource, click **File > Save** or click **Ctrl+S**, or click the **Save** button.

Result

You have created a new empty project in the TIBCO BusinessEvents Studio Development perspective.

In the TIBCO BusinessEvents Studio Explorer, the root folder of the project is called FraudDetection. It has a set of project subfolders.
In addition you may see links for various diagrams: `FraudDetection.conceptview`, `FraudDetection.eventview` and `FraudDetection.projectview`. These are respectively concept model, event model, and project (or element) diagrams. Diagrams are created dynamically and are not saved. More diagrams are available for other purposes. They are UML compliant, with some exceptions. Some diagrams are created only when generated.

Then, you will begin to define your TIBCO BusinessEvents project by building a channel for information to enter the deployed application, and a destination for the application to listen to.

The order in which you build up the project is not fixed. For example, you might define the project ontology first.

Adding an HTTP Channel and Destination

In this task you configure an HTTP channel with one destination. The `AllOps` destination listens for messages that come from HTTP forms embedded in the project's `readme.html` file.

Example projects use the HTTP channel because it does not require use of any external software. If you have TIBCO ActiveMatrix BusinessWorks, TIBCO Enterprise Message Service, TIBCO Rendezvous, or other source for messages you can experiment with adding different types of channels.

Learning Points

What are channels and destinations?

Messages enter and leave the system through channels. You create destinations within a channel to define the message sources and sinks. Typically, *events* are created using data in incoming messages; outgoing messages are created using data from events. Later in the tutorial, you will set up the relationship between these destinations and the event types that they listen to by default.

Note that in this tutorial outbound messages are simply sent to the console, so there are no outbound destinations.

How are channels and destinations created?

You create channels and destinations at design-time, as explained below. When you are planning TIBCO BusinessEvents projects, you would consider the incoming and outgoing messages for your project, and then define the channels, destinations, and the corresponding event types — outbound events are transformed into appropriate messages, and inbound messages are transformed into events of a specified type.

Why Use Shared Resources?

Shared resources are generally used in channels to configure communication with some external system such as a database server or JMS server. You can configure the connection once and use it in multiple places. If some configuration has to be changed, you just have to change it in one place.

> ✓ **Tip:** It's a good idea to use global variables in shared resources so that projects can be quickly adapted to run in different environments.

More Information

See *TIBCO BusinessEvents Developer's Guide* for more information on channels and destinations.

Adding an HTTP Connection

For this example project, you will add an HTTP connection.

Procedure

---

1. Select the `SharedResources` folder and press **Ctrl+N**.

    You will see the Select a Wizard dialog. You could also get here y selecting **File > New > Other** .

2. Select **TIBCO Shared Resources > HTTP Connection** and click **Next**.

3. In the New HTTP Connection Wizard, name the connection HTTPConnection and click **Finish**.

    In a real world situation you would probably give the connection a more meaningful name. You will see the HTTP Connection dialog.

    > **ⓘ Note:** Resource names and directory names in the path to a resource cannot be any of the keywords or other words listed in Rule Language Grammar in *TIBCO BusinessEvents Developer's Guide*, and they cannot contain spaces.

4. In the Host field, enter localhost.

5. In the Port field, enter 8108. This is the port used in the `readme.html` for this example.

6. Save the resource (click the **Save** button in the toolbar) and close it.

    > **ⓘ Note:** If you change the port here, change the readme port too. If port `8108` is not free, use an available port in the 8000 range. You must also edit the `readme.html` that goes with the project.

Adding an HTTP Channel and Destination

For this example project, you will add an HTTP channel and destination.
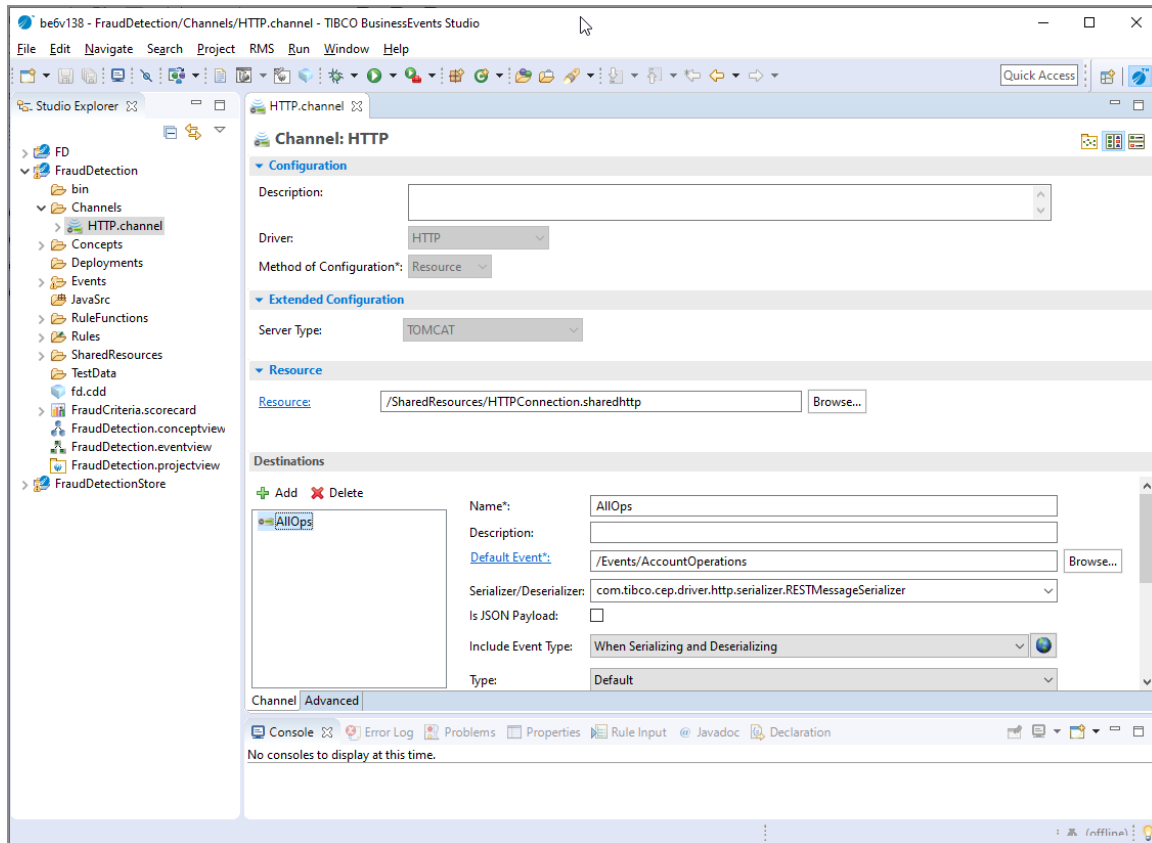
Procedure

1. Right-click the **Channels** folder and select **New > Channel**.

   The New Channel Wizard opens up.

2. Add information to the following fields:

   a. In the Channel Name field, type **HTTP**.

   > ⓘ **Note:** This value is case sensitive. Ensure that you use all capital letters.

   b. In the Description field, type **An HTTP channel**.

   c. In the Driver Type field, select **HTTP**.

   d. Click **Finish**.

      You will see the Channel editor.

> **ⓘ** **Note:** In the editor, you cannot change the resource name but you can update the description, if needed. You can later rename the resource using a shortcut menu refactoring option.

3. In the Resource field, browse to and select the HTTP connection resource you created.

   Only valid shared resources for the current resource display.

4. In the Destinations section, click **Add** and name the destination **AllOps**. Leave all other fields set to their default values.

> **ⓘ** **Note:** Provided Examples Requirements:
>
> - `HTTP` and `AllOps` are required names for examples. The readme.html uses the `AllOps` destination in its embedded forms.
> - The HTTP connection port must match the readme port .

5. Save and close the resource.

> **✓** **Tip:** All messages arriving at a destination are transformed to the destination's default event, unless the message specifies a different event. Adding a default event is not necessary for example projects, because the `readme.html`, which starts an HTTP channel, specifies the event to use.
>
> Adding a default event, however, does stop a warning sign from appearing. This warning draws your attention to the fact that a destination does not have a default event, but does not indicate an error.
>
> Default events and destinations are explained in Events.

What to do next

Now you have built a channel and a destination within that channel to listen for messages. The next step is to create some events.

Events

In this task, you begin to build the project ontology by defining some events — or strictly speaking, event *types*. Before you define event types in a real-world project, you first

examine the incoming and outgoing messages, as well as messages that you want to occur within the application, and configure each event type's characteristics accordingly. You can use inheritance (as demonstrated here) to simplify configuration.

TIBCO BusinessEvents provides various kinds of events. Simple events are used in this tutorial to bring messages into the application. In addition you can use SOAP events, time events and advisory events, which you can learn about in the product documentation.

Learning Points

What is an event?

The term *event* is overloaded: it means an activity that happens, and the definition of an object that represents the activity in TIBCO BusinessEvents (an event type), and an instance of that event type definition.

How are events (event instances) created?

Simple event types are created at design time. Event instances are generally created using data in incoming messages. When a destination receives a message, it creates an event to hold the information from the message. Events from channels are automatically asserted into the Rete network, where their presence generally triggers rules (if all rule conditions are met).

Simple events can also be created by rules and rule functions. Events created this way are not asserted automatically because they could be intended for use as outbound events, to be sent to a destination. You must explicitly assert such internally created events as needed.

What is an event payload?

Just as messages have properties and a message body, events can have properties and payloads. The payload is optional. It is used to hold more complex data, for example, SOAP messages. (The events in this example do not use a payload.)

What is a default destination?

Outbound events of the same event type are often sent to the same destination. To simplify the process of sending those events, you can specify a default destination in the event type.

What is a default event?

The default event configured for a destination is used to hold information transferred from an incoming message, when no event type is specified in the message.

See Default Destinations and Default Events in TIBCO BusinessEvents Architect's Guide for more details.

Why do the events have a warning sign?
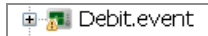
Just as destinations generally have default events, events generally have default destinations. The warning is to alert you to the fact that an event has no default destination. In this case, however, no events are sent out through channels so no default is required.

Rules that apply to a parent type also apply to its child types

Concept and event types use inheritance in a similar way to Java classes. Because *rules that apply to a parent type also apply to its child types*, it is generally not advisable to create many levels of inheritance. However it can be a useful technique. In this tutorial the `AccountOperations` event is the parent of both the `CreateAccount` and `Debit` events. You'll see why in a later section.

More Information

- See Channels and Events in TIBCO BusinessEvents Architect's Guide for overview and conceptual information.

- See Working with Events, Working With Time Events, and Advisory Events in TIBCO BusinessEvents Developer's Guide for implementation details.

Defining the AccountOperations Event

This event is a parent to events that are used in the project. It has one property: `AccountId`. All its child events inherit this property, and extend the parent by adding more.

Procedure

1. Right-click the **Events** folder, and select **New > Simple Event**.

   You will see the New Simple Event Wizard

2. In the Simple Event Name field, type **AccountOperations**.

3. In the Description field, type **Parent event for all account-related events**.

4. Click **Finish**

You will see the Simple Event Editor.

5. In the **Properties** section, click the **Add** button.

6. Click in the cell under Name and type the name **AccountId**

   It's a `String` property, and `String` is the default type.

7. Save and close the resource.

Defining the CreateAccount Event

For example purposes, the information needed to create an account is an ID, a balance, and a monthly average balance (for the fraud detection calculation). The ID property is inherited from the `AccountOperations` event.

Procedure

1. Right-click the **Events** folder, and select **New > Simple Event**.

   The New Simple Wizard appears.

2. In the Simple Event name, type **CreateAccount**.

3. In the Description field, type **Triggers the CreateAccount rule to create an account**.

4. Click **Finish**.

5. In the field Simple Event Editor Inherits From, click Browse.

6. In the upper section of the event picker, select the **Simple Event** event type, and in the lower section browse to and select the **AccountOperations** event. Click **OK**.

   The `AccountOperations` event is now the parent event for the `CreateAccount` one.

7. In the Default Destination field, click the browse button and in the Select Destination dialog, select **/Channels/HTTP.channel/AllOps**.

   Click **OK**.

8. In the **Properties** section, add two properties:

   - **Balance**, of type **double**
   - **AvgMonthlyBalance**, of type **double**

9. Save and close the resource.

> ✅ **Tip:** In any editor, you can click any label that is underlined (such as the Inherits from label in the Event editor) to open the resource selected for that setting.

Debit Event Reference

Values used for the `Debit` event.

Field	Value
Name (Defined in the wizard)	`Debit`
Description (Defined in the wizard and editor)	`Specifies an account (by inheritance) and a debit amount`
Inherits From	`AccountOperations`
Properties	`Amount` (double)

Unsuspend Event Reference

Values used for the `Unsuspend` event.

Field	Value
Name (Defined in the wizard)	`Unsuspend`
Description (Defined in the wizard and editor)	`Triggers the UnsuspendAccount rule to change the customer status from Suspended to Normal`
Inherits From	`AccountOperations`
Properties	(No properties)

What to do next

Next you will continue to configure the project ontology by defining a concept.

Account Concept

In this task, you define the Account concept, which holds basic information about an account: an ID, a balance, an average monthly balance, and an account status. You also learn some useful information about concepts and how they are used.

Learning Points

What is a concept?

A concept type is a definition of a set of properties that represent the data fields of an entity. Concept types are like Java classes, and concept instances are like Java objects.

How are concept instances created?

Concept instances are created by rules and rule functions. Information from event properties or payloads is often used to create concept instances, but other information can be used, for example, the results of a query or a calculation.

What is a database concept?

In TIBCO BusinessEvents, you can also create concepts by importing them from a database. A set of functions enables you to update the database record to account for changes made in TIBCO BusinessEvents. Unlike regular concepts, database concept

instances are not asserted to the Rete network automatically, and they do not track history.

How can I persist concept instances?

Instances of concepts (and events) are also known as *facts* and *entities*. They can be persisted in various ways, generally using a cache and backing store, as determined by the business need. Later tutorials explain these features.

How is history tracked?

When the History setting for a concept property is 0 (zero) the current value is stored without a date-time stamp. When the history setting is 1, the current value is stored, along with the date and time the value was added or changed. When the history value is greater than 1, TIBCO BusinessEvents tracks changes to property values up to the specified number (using a ring buffer). The Policy setting additionally determines what values are recorded, all values or only changes to the prior value.

You'll set the Debits property history, to track All Values, that is, TIBCO BusinessEvents records the value of the property every time an action sets the value, even if the new value is the same as the old value — a person can debit the account twice by the same amount. For a property such as *address* you might want to track only changes to the value.

More Information

- See Concepts in *TIBCO BusinessEvents Architect's Guide*.

- See Working with Concepts in *TIBCO BusinessEvents Developer's Guide*.

Defining the Account Concept

Procedure

1. Right-click the **Concepts** folder, and select **New > Concept**.

   You will see the New Concept Wizard

2. In the **Concept name** field, type `Account`. In the Description field, type `This concept maintains history for account transactions`.

3. Click **Finish**.

4. In the Account Concept Editor Properties section, add the following properties:

Name	Type	Policy	History
Balance	double	Changes Only	1
Debits	double	All Values	5
Status	String	Changes Only	1
AvgMonthlyBalance	double	Changes Only	0

   The Multiple field is used to define an array property. In this release, domains are

used only with the TIBCO BusinessEvents Decision Manager component.

You may wonder where the account ID from the incoming event will be stored. It will go into the concept's `extId` attribute.

> ℹ️ **Note:** Concept Attributes and Concept Relationships:
>
> **Attributes**
>
> Concepts, events, and scorecards have some built-in attributes, in addition to the properties you define here. The attribute `extId`, referenced as `@extId`, will hold the account ID.
>
> **The `extId` must be unique across the cluster**
>
> It's important to note that the optional `extId` attribute, if used, must be unique across all objects in the cluster. This attribute is used for events as well as for concepts. For example If you use database concepts you might expect to use the primary key from the database as the `extId`. However, that would not be possible if more than one table contains the same columns in its primary key.
>
> **Concept Relationships**
>
> Concepts can have containment and reference relationships with other concepts. You set these up as concept properties, and define the kind of relationship by selecting an appropriate data type for the property. For example, a car concept can contain Wheel concepts, and can have a reference relationship to a Dealership concept.
>
> **Inheritance**
>
> Concepts can also inherit from other concepts (and events can inherit from other events). Inheritance is a programming relationship and not an ontology relationship. It enables you to extend a base type for more efficient project management. .

5. Save and close the resource.

Result

You have defined a concept type to hold information about bank accounts. The last step in building the ontology of your project is to set up a scorecard to hold fraud detection criteria that are used in rules.

FraudCriteria Scorecard

In this task, you finish building the project ontology by creating a scorecard. The `FraudCriteria` scorecard will store the criteria used to determine fraud, not any specific data about customer accounts. In this example, you will use this scorecard in rules.

Learning Points

What is a scorecard?

> A scorecard is a special type of concept. A scorecard serves as a static variable. You can use a scorecard resource to track key performance indicators or any other information. Unlike concepts, there is only one instance of a scorecard. You create the scorecard at design time. Its values can be viewed and updated using rules.

> It is more accurate to say there is one instance of a scorecard per inference agent. This tutorial uses one inference agent. However, in the next tutorial you will deploy multiple agents, and each has its own instance of the scorecard. This enables scorecards to be used for local purposes and minimizes contention between the agents. Do not use scorecards as a mechanism to share data between multiple agents.

More Information

Look at Scorecards in *TIBCO BusinessEvents Developer's Guide*.

Creating the FraudCriteria Scorecard

There is only one scorecard in this project, so you do not need a folder for it. You will create it in the project root.

Procedure

1. Select the FraudDetection root folder and press `Ctrl+N`. In the Select a Wizard dialog, expand **TIBCO BusinessEvents** and click select **Scorecard**.

    This is how you add resources that are less commonly used.

2. Name the scorecard `FraudCriteria`.

    Add the description `Stores the criteria used to determine fraud`, and click **Finish** .

3. In the FraudCriteria Scorecard editor, add the following properties:

Name	Type	Policy	History
`interval`	long	Changes Only	0
`num_txns`	int	Changes Only	0
`debits_percent`	double	Changes Only	0

4. Save and close the resource.

Result

You have now set up the ontology for the project, that is, the definitions of all the events, concepts, and scorecards that are needed to store information (facts) about possible fraud detection.

What to do next

Next, you will configure a rule function that sets values for the scorecard, and one that acts as an *event preprocessor* — a term covered in the section Adding the PreProcessor Rule Function.

The InitializeScorecard Rule Function

In this task, you configure a rule function that initializes values for the `FraudCriteria` scorecard. This rule function is used at system startup. You can configure that connection later.

Learning Points

What is a rule function?

A rule function is a function you write in the TIBCO BusinessEvents rule language.

How are rule functions created and used?

You write rule functions using the rule editor. You can use rule functions in rules and other rule functions, in event preprocessors (which are explained in the section Adding the PreProcessor Rule Function), and as startup or shutdown functions for an agent.

What other types of functions are there?

TIBCO BusinessEvents provides a large library of functions for various purposes. In addition, an ontology function is automatically created for each concept and event in your project. These are constructor functions. Another type of ontology function enables you to create and schedule a time event. You can also add custom Java functions.

More Information

- Read about Rules and Functions in *TIBCO BusinessEvents Architect's Guide*.

- Read about Rules, Rule Templates, Functions, Rule Language Grammar, Mapping and Transforming Data, and XPath Formula Builder in *TIBCO BusinessEvents Developer's Guide*.

Adding the InitializeScorecard Rule Function

You will add a rule function, which is written in the TIBCO BusinessEvents rule language.

Procedure

1. Right-click the **RuleFunctions** folder, and select **New > Rule Function**.

   You will see the New Rule Function Wizard.

2. In the Rule Function name field, type **InitializeScorecard**.

   In the Description field, type **Sets values in the FraudCriteria scorecard**.

3. Click **Finish**.

   You can work in the Source view or the Form view, according to your preference. The tutorial uses the Form view. In the lower area, click the Form tab to switch to the Form view.
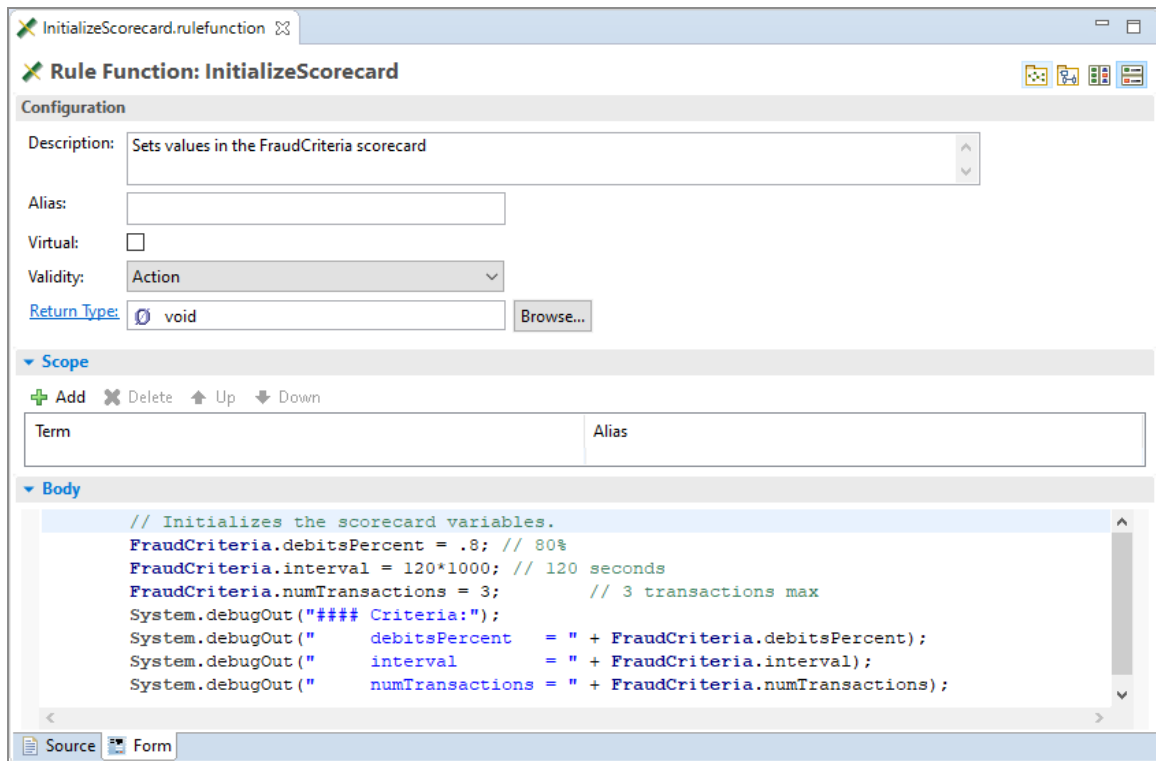
> ### Tip:
> **Rule Function Editor Preference**
>
> To set the default mode, go to **Window > Preferences > TIBCO BusinessEvents > Rules** and check or uncheck the following checkbox as desired: **Initially show "Form" tab in Rule Function Editor**.
>
> Ctrl-click the name of a rule function in a rule or rule function editor to open the editor for that rule function.

4. Leave the Scope area empty, because this function is used at startup. In the Body area, add the following lines to provide values to the `FraudCriteria` scorecard (and to comment your code):

```
//Initialize scorecard variables
FraudCriteria.debits_percent =.8;
FraudCriteria.interval       = 120*1000; /* 120 seconds */
FraudCriteria.num_txns = 3;
```

Notice that when you type the period after `FraudCriteria`, a list of its properties

appears so you can select a property.

5. Save and close the resource.

What to do next

You've set up a startup rule function. Next you'll set up an event preprocessor rule function.

PreProcessor Rule Function

HTTP is a request-reply protocol, and this step is required so that the HTTP server is ready to process the next request from the `readme.html` form. This rule function executes when an event is received. You can configure that connection later.

Learning Points

What is an event preprocessor?

An event preprocessor is a rule function that processes incoming messages before TIBCO BusinessEvents transforms them into events. In this case the preprocessor is used to send a response to the HTTP server. In real-world applications, however, a preprocessor might filter the messages so that only certain ones are used as events, and it might do other event enrichment actions. Preprocessors are multi-threaded and you can choose from various threading and queue options, as appropriate to handle the workload. By default the threading uses the system-wide shared queue and threads. See the topic Event Preprocessors in *TIBCO BusinessEvents Architect's Guide*.

How are event preprocessors configured for use?

A preprocessor is associated with a destination. It processes events arriving at that destination. See Cluster Deployment Descriptor (CDD) and the EAR File for details.

> **Note:** Locking is used to ensure that multiple concurrent RTCs (whether in the same agent or different agents in a cache cluster) do not work with the same object at the same time, or read an out of date version of the object. Locking is usually done in event preprocessors. See *TIBCO BusinessEvents Architect's Guide* to understand how to use locking to ensure data integrity within and across agents.

More Information

- See Event Preprocessors in *TIBCO BusinessEvents Architect's Guide*.

- See Event Preprocessors in *TIBCO BusinessEvents Developer's Guide*.

Adding the PreProcessor Rule Function

In this task, you configure a rule function that replies to the request received from the HTTP channel.

Procedure

1. Right-click the RuleFunctions folder, and select **New > Rule Function**.

   You will see the New Rule Function Wizard.

2. In the Rule Function name field, type **PreProcessor**.

   In the **Description** field, type `Closes requests from the HTTP server`.

3. Click **Finish**.

4. Click the **Form** tab at the bottom of the editor.

5. In the Scope section, click **Add**.

   You will see the Select Rule Function Scope Arguments dialog.

6. In the Types area (at the top), select **Event**. Click **OK**.

   In the Select Resource area, you could select a specific event type in the project. However, here we want any event to be in the scope of this rule function, not one specific event type.

7. In the Alias column (in the Scope section), replace the default alias (e) with **request**.

8. In the Body area, type: **Event.** (Event followed by a period).

   Notice that when you type the period (.) you see a list of all catalog functions in the Event category. Use the down arrow to scroll down the list of functions and stop at **replyEvent**. Its tooltip displays. Documentation for all catalog functions is provided in tooltips.

```
Function:        Event.replyEvent
Signature:       boolean replyEvent(SimpleEvent request, SimpleEvent reply)
                 Sends a SimpleEvent to the Destination specified within the request SimpleEvent such as an inbox in case of RV or a JMSReplyTo
Description:     property in case of JMS. The Destination in turn specifies the Channel where the SimpleEvent will be sent in the JMS Message. If the
                 original SimpleEvent does not have a reply destination or a subject set, this method will return false.
Parameters:
Name             Type              Description
request          SimpleEvent       The original request SimpleEvent.
reply            SimpleEvent       The Reply SimpleEvent.
Returns:         boolean           true if the SimpleEvent is sent; false otherwise.
Cautions:
See also:
Example:
```

The tooltips are also reproduced in the HTML version of the product documentation, in the Online References area.

9. Click the **replyEvent** function to select it.

   Now the body looks like this:

   ```
   Event.replyEvent()
   ```

   As you can see, the rule function arguments are two events, a request event and a reply event.

10. To specify the request event, type request , which is the alias for the scope argument you added.

11. To specify the reply event, just type request again. The reply event can be any event in this case, so we can simply reply with the request event.

What to do next

You have configured a rule function that will send a reply to requests sent by the HTTP server (through the HTTP channel). Next you will configure rules that take action on assertion of CreateAccount and Debit events, depending on various conditions.

BadCreateAccount and CreateAccount Rules

In this task you create two rules, one called CreateAccount and one called BadCreateAccount.

Both rules can fire when a CreateAccount event is asserted into the Rete network. However, the BadCreateAccount rule has a higher priority, so it will fire before the CreateAccount rule.

The `BadCreateAccount` rule checks whether the account ID provided in the `CreateAccount` event matches the account ID of any Account instance already in the Rete network. One of the two following situations must occur:

- A matching ID exists in the Rete network: the `BadCreateAccount` rule prints a message to the console, and "consumes" — that is, deletes — the `CreateAccount` event. Because that event is consumed, the `CreateAccount` rule cannot fire.

- No matching ID exists in the Rete network: the `BadCreateAccount` rule does nothing, and then the `CreateAccount` rule fires, and creates the `Account` concept instance.

Learning Points

What are rules and how are they created?

Rules define actions to take when certain conditions are met. Rules are written in the TIBCO BusinessEvents rule language, which is similar to the Java language. Rules are declarative and are generally narrow in scope. A rule has three parts: the declaration (`declare`), the conditions (`when`), and the actions (`then`). As with rule functions, you can work in a source view, which displays the Java-like code, or in a form view.

How are rules used at runtime?

The rule engine checks all changes and additions to the Rete network and evaluates or reevaluates rules, using their declaration and conditions, as needed. Eligible rules are added to the *rule agenda*.

What is the rule agenda

A rule fires when it is at the top of the agenda. The engine determines the order of firing using rule declarations and conditions, and each rule's priority and rank (if these features are used). As the contents of the Rete network change, the engine reevaluates rules and removes any that are no longer eligible to fire. See Understanding Conflict Resolution and Run to Completion Cycles in *TIBCO BusinessEvents Architect's Guide* for details.

How can you prioritize rule execution?

The Priority setting is used by the runtime engine when determining the order in which rules are fired. Those with a number closer to one fire first. Within a set of rules that has the same priority, a ranking feature enables you to determine which fire before others. It uses a callback rule function that enables you to specify business logic to establish the rank. When there is no reason to force rules to execute in a particular order, leave the Priority and Rank fields set to the default and let the runtime engine determine rule order.

How can you create concept instances?

In this task you instantiate an object instance using its ontology function. Another way is using the `Instance.CreateInstance()` function. This function uses the XSLT mapper to map any of the scope variables (such as properties, attributes and event payloads) to the new instance properties. You can also create event instances in a similar way, using the `Event.CreateEvent()` function.

More Information

- See all references provided for Adding the InitializeScorecard Rule Function.

- Read about Runtime Inferencing Behavior in *TIBCO BusinessEvents Architect's Guide*.

Adding the BadCreateAccount Rule

The `BadCreateAccount` rule has a higher priority, so it will fire before the `CreateAccount` rule.

Procedure

1. Right-click the **Rules** folder, and select **New > Rule**.

   You will see the New Rule Wizard.

2. In the Rule name field, type **BadCreateAccount**.

   In the **Description** field, type `Checks for an existing account with the specified ID`.

3. Click **Finish**.

4. In the **Form** tab, set the Priority field to `3`.

   This is a higher priority than 5, the default. You'll set the `CreateAccount` rule to priority 5, so that the `BadCreateAccount` rule always fires before `CreateAccount` rule.

5. Expand the Declaration section as needed so you can see empty rows below the headings Term and Alias.

   Each of the sections can be expanded and contracted as needed.

6. Drag the Account concept from the TIBCO BusinessEvents Studio Explorer tree into the first empty row in the Declaration section.

   You will see the project path of the Account concept in the Term column, and account in the Alias column.

7. Similarly, drag the `CreateAccount` event into the next available row.

   Declaration

   The Declaration provides the scope of the rule. It lists all the entity types to be used in the rule, and their aliases. By default the alias is set to the entity name in lower case letters. You can change it as desired.

8. In the Conditions panel, type the following:

   ```
   //Checks whether the extId of an Account instance in working memory
   //matches the incoming event's account ID
   account@extId == createaccount.AccountId;
   ```

   Notice that when you type the At sign (@), a pick list of concept attributes appears. Attributes are built-in. You cannot add or remove attributes. The `id` attribute value is set internally. However you can set the external ID, `extId`.

9. In the Actions panel, just below the Conditions panel, type these statements:

   ```
   System.debugOut("#### Account already exists: " +
   createaccount.AccountId);
   Event.consumeEvent(createaccount);
   ```

   These actions are done only if the conditions are met. If not, then the next rule in the agenda fires — and that is likely to be the `CreateAccount` rule, which you'll define next.

   You are also consuming the event, so it cannot trigger any other rules.

10. Save and close the resource.

Adding the CreateAccount Rule

The `BadCreateAccount` rule has a higher priority, so it will fire before the `CreateAccount` rule.

Procedure

1. Right-click the Rules folder again, and select **New > Rule**.

2. In the **New Rule Wizard Rule name** field, type `CreateAccount`.

3. Click **Finish**.

4. In the **Form** tab, set the **Priority** field to 5. (This is a lower priority than the BadCreateAccount rule.)

5. Drag the CreateAccount event from the TIBCO BusinessEvents Studio Explorer tree into the first empty row in the Declaration section.

   This rule has no conditions — the BadCreateAccount rule means there is no need. You could have combined the two rules into one. There are many ways to write rules for a project. You have to use your judgment.

   In the Actions panel, you'll use an ontology function to create the Account concept instance. You can also get to ontology functions using the function catalog.

6. From the top menu select **Window > Show View > Other > TIBCO BusinessEvents > Catalog Functions**.

   The Catalog Functions view displays on the right (unless your Eclipse IDE is configured differently — it could display along the bottom, for example).

7. Below each concept, event, and rule function is its ontology function. Expand **Ontology Functions > FraudDetection > Concepts > Account > Account**:
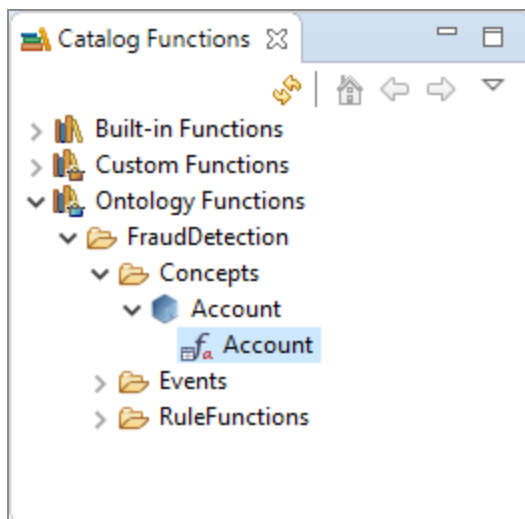
8. Drag the Account function into the Actions section.

   You see its signature:

```
Concepts.Account.Account(/*extId String */,/*Balance double
*/,/*Debits double */,/*Status String */,/*AvgMonthlyBalance double
*/)
```

9. Configure the function to create the Account concept, and type the rest of the actions as follows. You can double-click the tab to expand the rule editor, to make rule writing easier.

```
    Concepts.Account.Account(createaccount.AccountId/*extId String
*/,
            createaccount.Balance/*Balance double */,
            0/*Debits double */,
            "Normal"/*Status String */,
            createaccount.AvgMonthlyBalance/*AvgMonthlyBalance
double */);
    Event.consumeEvent(createaccount);
    System.debugOut("#### Created account " +
createaccount.AccountId);
```

10. Save and close the resource.

Note that in this case it is not really necessary to consume the event. No other rules have this event in their scope.

Result

Events with time to live (TTL) set to zero (0) are consumed at the end of an RTC. However, it does no harm and makes the rules more consistent, reducing chances of error.

ApplyDebit BadApplyDebit and CheckNegativeBalance Rules

It's important to understand how conflict resolution and run to completion (RTC) cycles work.

When a `Debit` event is asserted into the Rete network, TIBCO BusinessEvents checks rules with the Debit event in their scope.

- The `ApplyDebit` rule has the Debit event in its scope, and also an account ID. It's priority 1 so it will execute before any other lower priority rule. The engine checks the rule conditions.

  - If the Rete network also contains an Account instance whose ID matches the ID in the debit event, the rule executes. If the account is suspended, a message to that effect displays in the console. Otherwise, the account is debited. A

message displays in the console to this effect.

- ○ If the Rete network does not contain a matching Account instance, then other rules in the agenda — those that have debit events in their scope — are eligible to execute. `BadApplyDebit` is the only rule with the Debit event in its scope, so it is eligible and it executes. It sends a message to the console that no matching account is found.

- The `CheckNegativeBalance` rule has an `Account` concept in its scope. When an account is debited the Account concept is changed, and so the `CheckNegativeBalance` rule becomes "newly true." The effect is similar to assertion of a new entity into the Rete Network. If the debit has made the account balance negative, this rule sets the status to Suspended.

Learning Points

If you understand what triggers rules to execute, and why a rule may not execute, you can design rules more effectively.

For a reminder, carefully reread Understanding Conflict Resolution and Run to Completion Cycles in *TIBCO BusinessEvents Architect's Guide*.

More Information

- See all references provided for Adding the InitializeScorecard Rule Function.

- See Runtime Inferencing Behavior in *TIBCO BusinessEvents Architect's Guide*.

Rules Examples

This section shows the source code for the three rules you will add.

If you have completed Adding the BadCreateAccount Rule and Adding the CreateAccount Rule, you will be familiar with adding rules.

In addition, a screenshot of the `ApplyDebit` rule is shown below.

ApplyDebit Rule Form View

The main purpose for showing the form view is so you can compare it with the source view when creating your own rule.
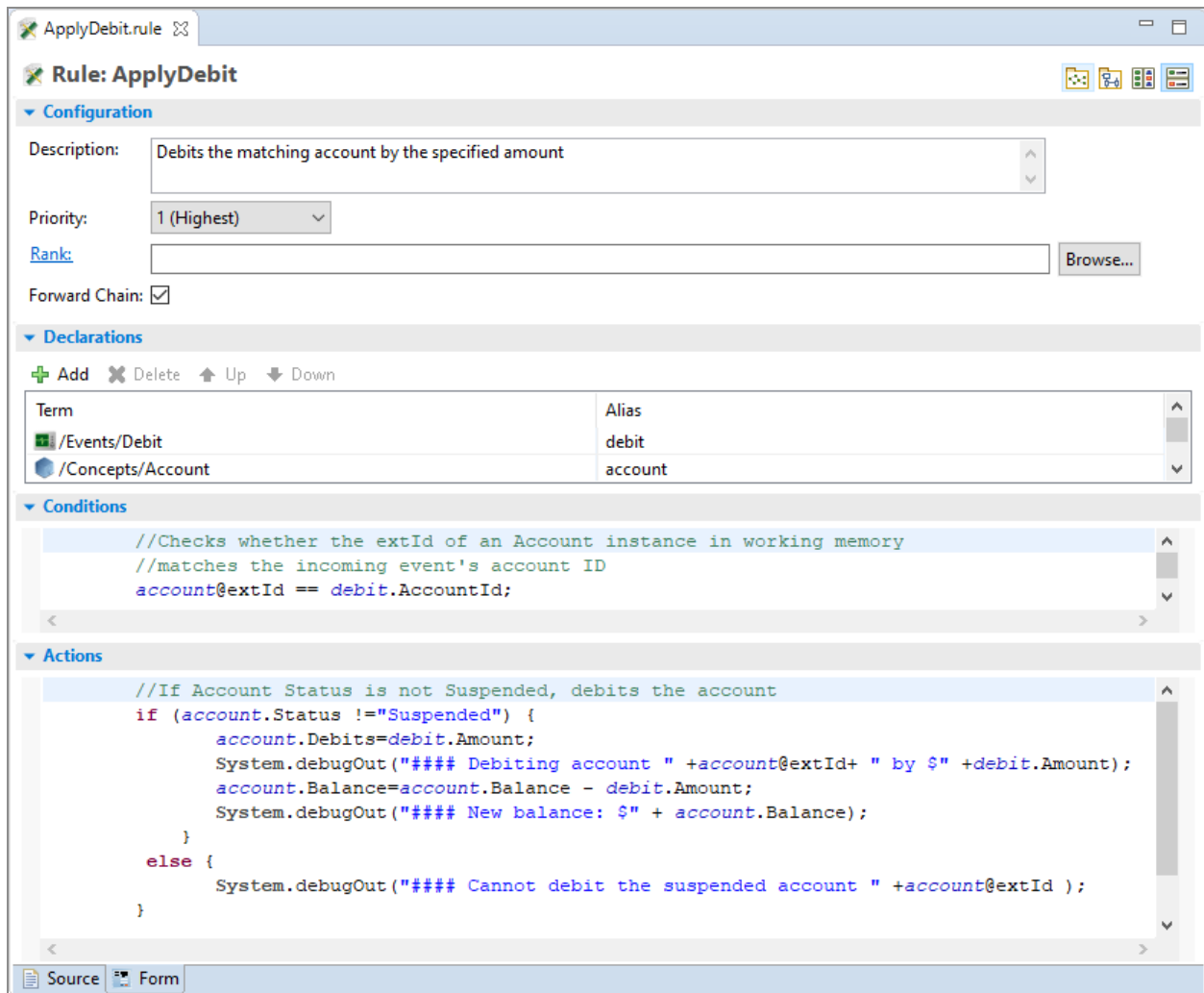
ApplyDebit Rule Source View Code

```
/**
 * @description Debits the matching account by the specified amount
 * @author
 */
rule Rules.ApplyDebit {
        attribute {
                priority = 1;
                forwardChain = true;
        }
        declare {
                Events.Debit     debit;
                Concepts.Account         account;

        }
```

```
        when {
                //Checks whether the extId of an Account instance in working memory
                //matches the incoming event's account ID
                account@extId == debit.AccountId;
        }
        then {
                //If Account Status is not Suspended, debits the account
                if (account.Status !="Suspended") {
                        account.Debits=debit.Amount;
                        System.debugOut("#### Debiting account " +account@extId+ " by
$" +debit.Amount);

                        account.Balance=account.Balance - debit.Amount;
                        System.debugOut("#### New balance: $" + account.Balance);
                    }
                 else {
                        System.debugOut("#### Cannot debit the suspended account "
+account@extId );
                }

                Event.consumeEvent(debit);
        }
}
```

BadApplyDebit Rule Source View Code

```
/**
 * @description
 * @author
 */
rule Rules.BadApplyDebit {

        attribute {
                priority = 5;
                forwardChain = true;
        }

        declare {
                Events.Debit    debit;
        }

        when {
        }

        then {
                System.debugOut("#### Debit not applied, account not found: " +
debit.AccountId);
```

```
                    Event.consumeEvent(debit);
        }

  }
```

CheckNegativeBalance Rule Source View Code

CheckNegativeBalance Rule Source Code

```
/**

 * @description

 * @author

 */

rule Rules.CheckNegativeBalance {


    attribute {

        priority = 5;

        forwardChain = true;

    }


    declare {

        Concepts.Account account;

    }
```

```
    when {

        // Checks that the balance is less than zero

        account.Balance < 0;

        // Checks that Account status is not set to
Suspended

        account.Status != "Suspended";

    }


    then {

        account.Status = "Suspended";

        System.debugOut("#### Account ID " +
account@extId

            + " STATUS set to Suspended. Balance "

            + account.Balance + " is less than zero");

    }


}
```

FraudDetection Rule and Unsuspend Account Rule

Add the `FraudDetection` rule, typing in the Source view or Form view, according to your preference.

Most of the code is in the conditions. The first condition checks whether the number of debits in the specified interval prior to the current time is greater than the specified number of debits. The interval and the number of debits are set in the `FraudCriteria` scorecard.

The second condition checks whether the sum of all debits in the verification interval is greater than the specified percentage of the account average monthly balance. The specified percentage is set in the `FraudCriteria` scorecard. The average monthly balance, for the purposes of this tutorial, is set in the Account instance created by the `CreateAccount` rule.

You will also add a rule called `UnsuspendAccount`. This is a convenience rule that allows you to change a customer status from Suspended to Normal at run-time.

Learning Points

How are conditions processed?

All conditions in a rule must be met, before the action is done. That is, each condition is joined by an implied AND operator.

In what order are conditions evaluated?

To learn more the effect of filters, equivalent join conditions, and non-equivalent join conditions on the efficiency of a rule, see Order of Evaluation of Rule Conditions in *TIBCO BusinessEvents Architect's Guide*. Understanding these points helps you design an efficient project.

How can I learn about all these catalog functions?

TIBCO BusinessEvents provides hundreds of catalog functions for use in rules and rule functions. You can use functions you already know about by typing the beginning of the name and then using the completion hints that appear. To learn about more functions, you can open the Catalog Functions view and browse. To see the tooltip for a function, hover the mouse over the function name. You can then drag a function into the editor, as you did in Adding the CreateAccount Rule. As a reminder, here's how to open the Catalog Functions view: **Window > Show View > Other > TIBCO BusinessEvents > Catalog Functions**. The tooltips are also available in HTML form, in the Online References section of the HTML documentation for the product.

More Information

- See all references provided for Adding the InitializeScorecard Rule Function

- Read about Runtime Inferencing Behavior in *TIBCO BusinessEvents Architect's Guide*.

Adding the FraudDetection Rule

Here is the example code for adding the FraudDetection rule.

```
/**
 * @description
 * @author
 */
rule Rules.FraudDetection {

        attribute {
                priority = 5;
                forwardChain = true;
        }

        declare {
                Concepts.Account        account;
        }

        when {

                //1. Checks that the Account status is not set to Suspended
                account.Status != "Suspended";

                //2. Checks the number of debits in the verification interval
                (Temporal.History.howMany(
                        account.Debits,
                        DateTime.getTimeInMillis(DateTime.now()) -
FraudCriteria.interval,
                        DateTime.getTimeInMillis(DateTime.now()),
                        true)
                   > FraudCriteria.numTransactions &&

                //3. Checks the percentage of the average balance that was debited in
the verification interval
                Temporal.Numeric.addAllHistoryDouble(
                        account.Debits,
                        DateTime.getTimeInMillis(DateTime.now()) -
FraudCriteria.interval)
                   > (FraudCriteria.debitsPercent * account.AvgMonthlyBalance));
```

```
        }

        then {
                account.Status = "Suspended";
                System.debugOut("#### Account ID " + account@extId
                        + " STATUS set to Suspended. Fraud suspected.");
        }

}
```

Adding the UnsuspendAccount Rule

Here is the sample code for adding the UnsuspendAccount rule.

```
/**
 * @description
 * @author nprade-nb
 */
rule Rules.UnsuspendAccount {

        attribute {
                priority = 1;
                forwardChain = true;
        }

        declare {
                Concepts.Account account;
                Events.Unsuspend request;
        }

        when {
                account@extId == request.AccountId;
                account.Status == "Suspended";
        }

        then {
                account.Status = "Normal";
        }

}
```

Congratulations! You have now configured the project's ontology and rules. Now you are ready to configure the Cluster Deployment Descriptor (CDD) and build the archive for deployment. But before you do, it's wise to validate and analyze the project, and look at it

in the Project diagram.

Project Analysis and Validation

In this task, you check the project for errors.

If you have configured it correctly, you'll see warnings but no errors. Warnings indicate potential issues, such as absence of default events. It's always worth checking them. In the FraudDetection project, the warnings do not indicate any actual issues.

Learning points

How can I validate that there are no basic errors in my code?

To perform checks that do not take project logic into account, use Project > Validate Project (or right-click on a project name, and use the same named option).

What if there are problems in my code?

You can debug it. TIBCO BusinessEvents builds on the Eclipse debugger features to provide additional checks. Developers familiar with the Eclipse debugger will find it intuitive to use.

How can I analyze that the logic of my project is OK?

For this kind of checking you use Project Analyzer or the Element Diagram or both. They work together to provide insights into your project. Project Analyzer is a document generated in the Problems view area. Element Diagram provides a visualization of the whole project or of selected project elements. It also shows all dependencies in a project. Project Analyzer and Element Diagram can be configured to run separately or together, using preferences.

How can I visualize different aspects of a project?

TIBCO BusinessEvents provides diagrams similar to UML sequence diagrams, class diagrams, and dependency diagrams.

How can I change the names of project resources, or move them around? Use the refactoring features. They ensure that changes you make are reflected throughout the project (with certain limitations).

More Information

In *TIBCO BusinessEvents Developer's Guide*, see Testing and Debugging Projects.

Analyzing and Validating the Project

Here is how you perform analysis and validation of a project.

Procedure

1. In TIBCO BusinessEvents Studio Explorer, highlight the project name, then from the top menus select **Project > View**.
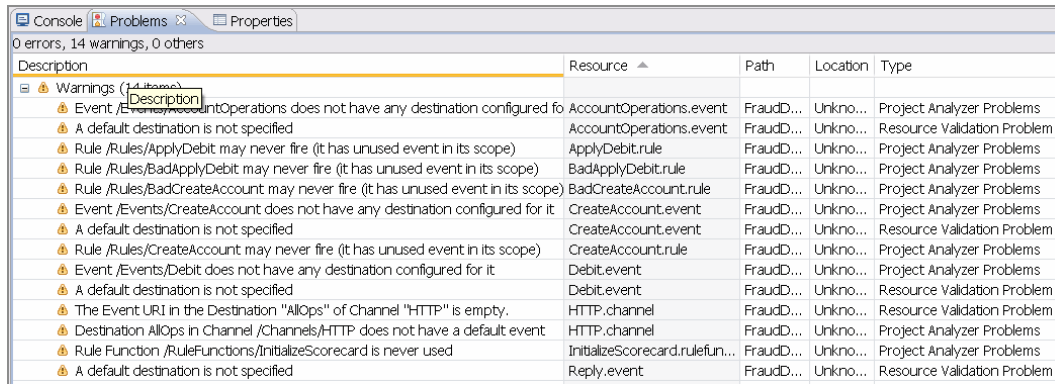
    You will see the project visualization in the main editor window.
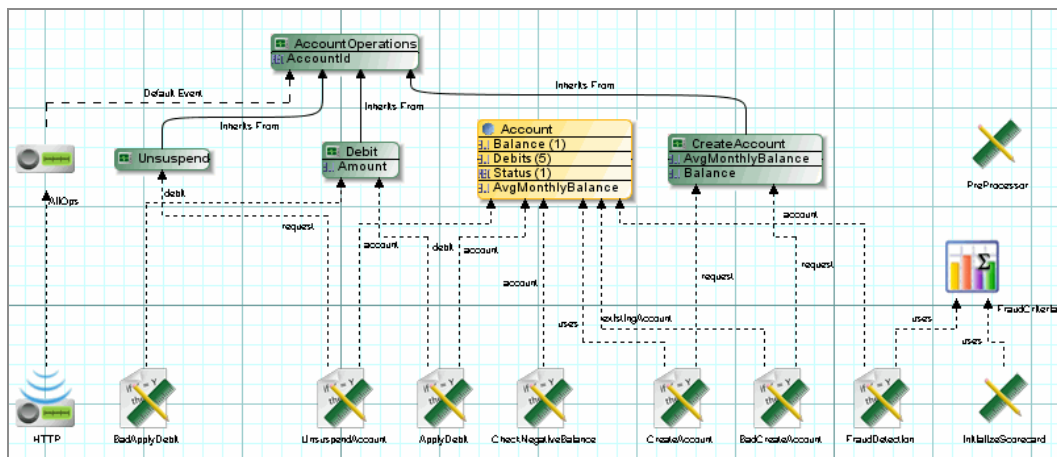
    You can see the dependencies and relationships between the project resources. You can jump to a resource editor by clicking the resource's icon. You can use the Project Filter palette which appears to the right of the diagram to show selected types of items only.

2. Click the **Problems** tab to see the project analysis.

    The Type columns shows the type of check that resulted in the warning. You can sort by any column. Here are some warnings about actions taken during project development. They are sorted by Resource and some columns are sized smaller to make others more visible.

Description	Resource ▲	Path	Location	Type
⊟ ⚠ Warnings (14 items)				
⚠ Event /Events/AccountOperations does not have any destination configured fo	AccountOperations.event	FraudD...	Unkno...	Project Analyzer Problems
⚠ A default destination is not specified	AccountOperations.event	FraudD...	Unkno...	Resource Validation Problem
⚠ Rule /Rules/ApplyDebit may never fire (it has unused event in its scope)	ApplyDebit.rule	FraudD...	Unkno...	Project Analyzer Problems
⚠ Rule /Rules/BadApplyDebit may never fire (it has unused event in its scope)	BadApplyDebit.rule	FraudD...	Unkno...	Project Analyzer Problems
⚠ Rule /Rules/BadCreateAccount may never fire (it has unused event in its scope)	BadCreateAccount.rule	FraudD...	Unkno...	Project Analyzer Problems
⚠ Event /Events/CreateAccount does not have any destination configured for it	CreateAccount.event	FraudD...	Unkno...	Project Analyzer Problems
⚠ A default destination is not specified	CreateAccount.event	FraudD...	Unkno...	Resource Validation Problem
⚠ Rule /Rules/CreateAccount may never fire (it has unused event in its scope)	CreateAccount.rule	FraudD...	Unkno...	Project Analyzer Problems
⚠ Event /Events/Debit does not have any destination configured for it	Debit.event	FraudD...	Unkno...	Project Analyzer Problems
⚠ A default destination is not specified	Debit.event	FraudD...	Unkno...	Resource Validation Problem
⚠ The Event URI in the Destination "AllOps" of Channel "HTTP" is empty.	HTTP.channel	FraudD...	Unkno...	Resource Validation Problem
⚠ Destination AllOps in Channel /Channels/HTTP does not have a default event	HTTP.channel	FraudD...	Unkno...	Project Analyzer Problems
⚠ Rule Function /RuleFunctions/InitializeScorecard is never used	InitializeScorecard.rulefun...	FraudD...	Unkno...	Project Analyzer Problems
⚠ A default destination is not specified	Reply.event	FraudD...	Unkno...	Resource Validation Problem

3. Close the `FraudDetection.projectview` tab.

   Diagrams are not saved. They are generated when needed.

What to do next

Now you are ready to configure the Cluster Deployment Descriptor (CDD) and build the archive for deployment.

Cluster Deployment Descriptor (CDD) and the EAR File

To deploy a project you need a CDD file and an EAR file. The CDD is not included in the project EAR. This means you can reconfigure a project's deployment configuration without having to rebuild the EAR.

Learning points

What is an EAR

The Enterprise Archive or EAR file contains details for all the resources in a project, and project global variables.

What is a CDD

The project's deployment configuration is defined in an XML file called *Cluster Deployment Descriptor* or *CDD*. You edit this file using the TIBCO BusinessEvents Studio Cluster Deployment Descriptor editor.

How do I set up preprocessors and start and shut down rule functions?

The CDD is where you configure rule functions to act as event preprocessors, startup rule functions, or shutdown rule functions. Only rule functions whose Validity setting is `Action` are valid for these uses. (These rule functions cannot require anything to be in

their scope, because they execute outside of the context of the Rete network and TIBCO BusinessEvents project resources.) A preprocessor is associated with a destination. It processes events arriving at that destination.

What is an object management type?

Object management (OM) refers to how TIBCO BusinessEvents manages and uses the objects generated within the TIBCO BusinessEvents application at runtime. This tutorial focuses on the basics, so it uses In Memory, the simplest OM type. Objects are kept in memory only, and are lost when the engine stops. Cache OM is explained in Cache OM Tutorial and backing store for this Cache OM is explained in Backing Store Tutorial.

> ✅ **Tip:** Other ways to create memory-based projects is to use Cache OM or Store OM. For cache OM, you can set the "mode" to Memory Only on all objects. This method provides fault tolerance and other advantages. See Cache OM with Memory Only Mode on All Objects and Fault Tolerance in *TIBCO BusinessEvents Architect's Guide*.

What is an inference agent?

An agent does certain work within the engine. With Cache OM, different types of agents do different work. But for this simple In Memory OM project, you use only one agent of one type, and that is the inference agent. Inference agents listen for messages arriving at destinations, and transform them into events. The events trigger rules, using the agent's Rete network and forward chaining, and the inference agent executes the rules.

What is a processing unit?

A processing unit is deployed as a TIBCO BusinessEvents engine. It has one or more agents (depending on OM and requirements) and it runs in one JVM. For In Memory OM, you generally do not have to do much configuration of processing units. The default processing unit settings are usually sufficient.

More Information

- In *TIBCO BusinessEvents Developer's Guide*, see Agent and Processing Unit Configuration

- See also *TIBCO BusinessEvents Configuration Guide*.

Adding and Configuring a CDD

You can create multiple CDD files for a project and at deploy time use the one that has the configuration you want to use.

Procedure

1. In TIBCO BusinessEvents Studio Explorer, right-click the project name `FraudDetection` and select **New > Cluster Deployment Descriptor**.

   You will see the New Cluster Configuration Wizard.

2. In the **File** name field, type `fd` and click **Next**.

   Unlike in other project resources, you can change the name later as desired. Short names are easier to type when starting the engine at the command line.

3. In the Object Manager Selection page, select the following values and click **Finish**.

   - **Cluster Provider Type** - `Unclustered`

   - **Object Management Type** - `In Memory`

   You will see the CDD editor, displaying the template for In Memory OM. For an In Memory project, very little deployment configuration is required.

4. Click the **Collections** tab, and do the following:

   a. Select Input Destinations and click **Add**.

   b. In the Destinations Collection field, type Http and again click **Add**.

   c. In the Select Input Destinations dialog, select /Channels/HTTP/AllOps and click **OK**.

   A Configuration panel appears.

   d. In the Input Destination ID field, type AllOperations, replacing the generated ID.

   e. In the Preprocessor field, select /RuleFunctions/PreProcessor.

   > **ⓘ Note:** Now this preprocessor will act on events arriving at the AllOps destination.

   The Configuration panel looks like the follow image:
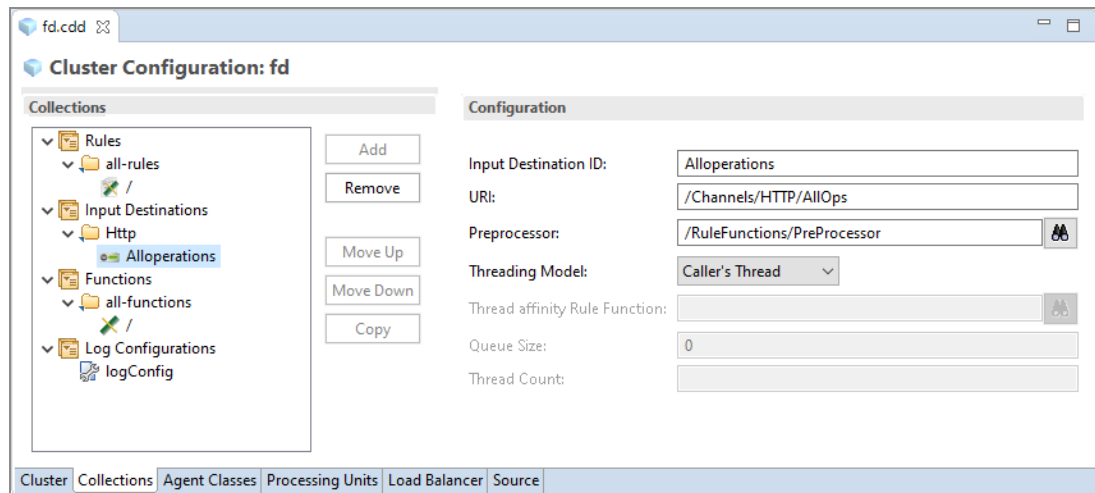
Collections enable you to create resources you can reuse when configuring multiple agent classes. Collections are used here to demonstrate the feature. In simple projects, you could simply configure the agent class without using collections.

5. Select the Agent Classes tab and expand the default agent class, which is called *inference-class*.

    When configuring an agent class, you can select a subset of the project rules, select and configure destinations, and select startup and shutdown rule functions. Thus different agent classes can behave quite differently at runtime.

6. Select Input Destination Collections and click **Add**.

    You see the Select Input Destinations dialog. In the Reference Collections area, select Http and click **OK**.

    As mentioned above, you can configure input destinations here, or link to input destinations configured in the Collections tab, or use both methods. Here we reference the collection you already defined.

7. Select **Startup Functions** and click **Add**.

    In the Select Rule Functions dialog, select /RuleFunctions/InitializeScorecard and click **OK**. When the engine starts, this rule function executes and initializes the scorecard values.

8. Save and close the CDD.

Building the EAR File

When you build the EAR file you must save it outside the project tree, or it will be recursively included in the next EAR file you build!

Procedure

1. In TIBCO BusinessEvents Studio Explorer, highlight the project name, then from the top menus select **Project > Build Enterprise Archive**.

   If you see a message asking you to save all project resources, click Yes. It means an unsaved resource editor is open.

   At the Build Enterprise Archive dialog, you can change the name to `fd`.

   > ✓ **Tip:** Short names are easier to type when starting the engine at the command line.

2. In the **File Location** field, browse to and select the directory above the project directory.

   To build the EAR in the provided example location, choose `BE_HOME`/examples/standard/FraudDetection/FD.ear.. Replace *BE_HOME* with your actual value.

   Where you build the EAR is not so important. You just have to specify the correct location when starting the engine at the command line.

3. Click **Apply**, then click **OK**. You see messages as the EAR file builds, then you see a message that the EAR file has built correctly:
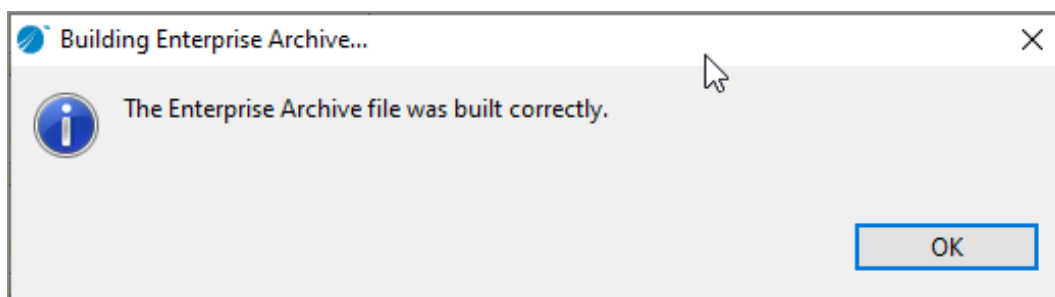
Result

Congratulations! You are ready to deploy the FraudDetection project.

Starting the Engine and Sending Events

For a detailed explanation of the above format see "Starting a TIBCO BusinessEvents Engine at the Command Line" in *TIBCO BusinessEvents Administration.*

The format for starting an engine at the command-line is as follows:

```
BE_HOME/bin/be-engine [-h] [--propFile
                            startup property file] [--propVar
                            varName=value][-p
                            custom property file]  [-n
                            engine name]  [-d]  -c
                            CDD file  -u
                            processing unit ID     EAR file
```

> **Note:** The _annotations.idx file requires write permissions for successful engine startup. By default, this file is created at BE_HOME/bin. If you do not have write permissions to this directory, configure the path for the _annotations.idx file using the be.engine.annotations.idx property in the be-engine.tra file.

Command Line Start

You can open a command window and at the command prompt and start the TIBCO BusinessEvents engine.

> **Tip:** You can simply open the example readme.html file and follow instructions there. In order to send events into the engines, you must enter information into the forms provided in the readme.
>
> Navigate to BE_HOME/examples/standard/FraudDetection and open readme.html in your browser. Follow the instructions in the readme file to start the engine and send events to the inference agent.

Open a command window and at the command prompt, start the TIBCO BusinessEvents engine using the following command. Locations specified are those of the provided configured example. Substitute your values for BE_HOME, and the location of the EAR and CDD files:

```
cd BE_HOME/examples/FraudDetection
BE_HOME/bin/be-engine --propFile BE_HOME\bin\be-engine.tra -u default -c
FraudDetection/fd.cdd fd.ear
```

It can be convenient to copy the CDD from your workspace to the same location where you saved the EAR file.

Congratulations!

You have completed the tutorial and now understand the basics of project design and deployment. Next you can undertake the tutorial provided in Cache OM Tutorial.

Cache OM Tutorial

In this tutorial, you will add caching functionality to your project and deploy two cache agents and one inference agent.

This tutorial uses the FraudDetection project configured in Project Design Tutorial.

Cache OM is used in most production systems because of the additional functionality, reliability, and high availability it provides, especially when a backing store is also used to store the data generated by the system on disk.

To explore basic caching features, you will make some minor modifications to the FraudDetection project used in the Project Design Tutorial. Modifications are required to retrieve objects from cache, or to check that objects do not already exist in cache before creating them.

Then you'll deploy two engines, each with one agent: an inference agent and a cache agent. You will observe how stopping and starting the inference agent does not affect how events are handled, because the objects are stored in the cache. You can also start a second cache agent to observe how two cache agents work together. You can stop and restart one cache agent with no loss of data.

Before you begin the tutorial, take a few minutes to learn a few important points about cache object management, in the next section, TIBCO BusinessEvents Cache Fundamentals. Setting up cache OM is fairly simple. Understanding how to manage cache objects and how to design for concurrency take some thought.

> **ⓘ Note:** An active LAN connection (device enabled and network cable plugged in) is required for Legacy ActiveSpaces cluster to work.

TIBCO BusinessEvents Cache Fundamentals

Read this section to grasp the basic ideas you need to understand and work with cache object management.

For greater depth, see *TIBCO BusinessEvents Architect's Guide*.

What is object management?

Object management (OM) refers to various ways that TIBCO BusinessEvents can manage the objects created by the actions of an inference agent's rules and rule functions. You define the OM options in the project's CDD file. The objects being managed of the types defined in the ontology, that is, concepts, scorecards and events. In the Project Design Tutorial, you used *In Memory OM*.

With In Memory OM, when an engine shuts down, all its data is lost. This option is useful for a project where objects are not created or do not have to be persisted, for example, one project that routes incoming events to other TIBCO BusinessEvents projects for handling.

What is a cache cluster?

When you use Cache OM, objects are persisted redundantly in memory caches. Each TIBCO BusinessEvents engine (JVM) participates as a node in the *cache cluster*. The cache manager manages the objects across the JVMs.

What is a distributed cache?

TIBCO BusinessEvents uses a pre-configured *distributed cache* scheme.

In a distributed cache, the cached object data is partitioned between cluster nodes (TIBCO BusinessEvents engines). Each object is stored redundantly in two or more nodes, for reliability and high availability: if one engine fails, the objects are re-partitioned across the remaining engines so that the configured number of backups is maintained.

What is a backing store?

With a distributed cache, one or more engines can fail without loss of data (depending on the number of backups of each object). However if all engines fail, the data is lost. For this reason it is common to write the objects to a database, known as the *backing store*.

TIBCO BusinessEvents provides tools to create the backing store for a project. When you use a backing store you can also use a *limited cache* and populate it as needed from the backing store, balancing performance and memory requirements as needed. (To learn how to set up the backing store, see Backing Store Tutorial.)

What are cache agents?

Cache agents store and serve the cache data for the cluster. They participate in distribution, partitioning and storage of the objects in the cluster.

An engine (processing unit) that runs a cache agent cannot run any other agents of any type. (For testing you can configure an inference agent to act also as a cache agent, but this is not recommended for production systems.)

What other types of agents are there?

Besides *cache agents* and *inference agents*, the *Query agents* can also join the cluster.

How do agents use cached data?

The inference agents can use objects in the cache as well as those they create from data coming in through channels (or create internally using rules and rule functions). Cached concept objects are shared by all agents in the cluster, scorecards are kept local to an agent, and events are clustered between the agents.

What are modes?

Each object type can use a different mode: *Cache Only*, *Store Only*, and *Memory Only*.

Cache-only is the default and that is the behavior described in the rest of this tutorial.

With Cache Only mode, how does data get from the Rete network to the cache?

At the end of each RTC, that is, each "run to completion" cycle, data is flushed from the Rete network to the cache. In this way the Rete network does not become clogged with irrelevant data. (This behavior occurs when the entities are configured to use cache-only mode, which is the default and is strongly recommended.)

With Cache Only mode, how does data get from the cache into the Rete network?

Before each RTC, you must load the relevant data into the Rete network from the cache. If you do not then cached data that is needed by the rules triggered by incoming events is not present. For example, a request to create a new account arrives through a channel.

It is important to check that no matching account already exists in the cache before you create the new account. To do so, you use an event preprocessor rule function to load any matching accounts into the Rete network. Then in the RTC rules can check for existing accounts and avoid creating duplicates.

How should I use Memory Only mode?

Use this mode for entity types whose data is transient and does not need to be persisted.

How do I change the number of backups of each object?

By default one backup of each cache object is maintained. Backups are maintained in different cache agents.

You can increase the number of backups by changing the Number of Backup Copies setting in the Object Management area of the CDD.

How does TIBCO BusinessEvents deal with concurrency and locking?

Multiple inference agents can run concurrently, sharing the same ontology and same cache cluster.

Another way to achieve concurrency is to use the multi-threaded Rete feature, which also requires cache OM. With agent or RTC concurrency, you must use locking: in both cases multiple RTCs are being processed at the same time, and data must be protected as in any concurrent system.

The tutorial does not demonstrate concurrency features. See *TIBCO BusinessEvents Architect's Guide* for more details.

FraudDetection Example

In this tutorial you will use a copy of the provided FraudDetection example project.

If you do not want to build the Fraud Detection Cache (`FraudDetectionCache`) project yourself, you can import the provided `FraudDetectionCache` example to see how it is configured.

You can also use the FraudDetection project configured in Project Design Tutorial. In that case, you will rename the project you configured earlier.

See Dependency of Interactive Readme File on a Higher Level Directory to understand how to organize your files so that you can send events to the running engine.

Learning points

What is refactoring?

When you make changes to the structure of a project element, such as renaming or moving the element, all references to that element are updated accordingly. This operation is known as *project refactoring*. Some refactoring type checks are also done for copy-paste operations.

More Information

- See Element Refactoring Operations in *TIBCO BusinessEvents Developer's Guide*.

Renaming the Fraud Detection Project and Changing Port

Procedure

1. Open TIBCO BusinessEvents Studio and, as needed, import the FraudDetection project (or a copy of it) into your workspace.

   See Import of Existing Projects into Your Workspace.

2. Highlight the project name and select **File > Refactor > Rename**.

   Type the name FraudDetectionCache.

3. Open the `HTTPConnection` resource that's in the `SharedResources` folder and change the port to `8109`. The FraudDetectionCache project ships with this port, and its file `readme.html`points to it as well (see Import of Existing Projects into Your Workspace for details).

4. Save these changes.

What to do next

Next you will configure the project to load objects from the cache into the Rete network before the start of an RTC.

Event Preprocessor and the Rete Network

Since this project uses Cache Only mode, after each RTC the data created in the agent is saved to the cache and removed from the Rete network.

Before the next RTC, therefore, you must reload any data needed by any rules that will be triggered in the RTC. This is generally done using the event preprocessor rule function, as it is the presence of event data that triggers most rules.

Learning Points

Loading data from cache into the Rete does not trigger rules

At least not in the way it would if the data was newly arriving in an event. In conjunction with other data, however, the presence of the loaded data can trigger rules, in the usual way.

More Information

In *TIBCO BusinessEvents Architect's Guide* see the following:

- Cache Modes and Project Design.

- Understanding Conflict Resolution and Run to Completion Cycles.

See also TIBCO BusinessEvents Cache Fundamentals to understand more about cache modes.

Updating the Event Preprocessor

Procedure

1. In TIBCO BusinessEvents Studio, open the **RuleFunctions > PreProcessor** rule function.

2. Click the **Source** tab at the bottom of the editor to work in the Source view.

   In the Project Design Tutorial you worked in the Form view. You could stay in that view, but it's good to become familiar with both views.

3. In the Scope section, replace `Event request` with `Events.AccountOperations request`. The section looks like the following:

   ```
   scope {
       Events.AccountOperations request;
   }
   ```

   The reason you have to narrow the scope is that the base Event class does not have an `AccountId` property, so it cannot be used in the function you'll add in the next step.

4. In the Body section, add two new lines (shown in bold), just before the closing bracket:

```
    body {
        // Replies to the request event, in order to close the HTTP
request.
        // To keep it simple, uses the request event as the response.
        Event.replyEvent(request, request);
        // Attempts to load any existing matching account.
        Cluster.DataGrid.CacheLoadConceptByExtId(request.AccountId,
false);
    }
```

The `Cluster.DataGrid.CacheLoadConceptByExtId()` function loads any matching items from the cache into the Rete network. In this case, it loads any concept whose `ExtId` matches the `AccountId` in the incoming event. The loading is done before the event is asserted, so the Rete network will contain any matching Account concepts. The `BadCreateAccount` rule then fires and as in the Project Design Tutorial, to prevent creation of duplicate accounts.

Result
You have modified the basic FraudDetection project to load instances from the cache into the Rete network using an event preprocessor. This enables the project code to check for existing instances first, when creating new ones.

What to do next
Next you will work in the Cluster Deployment Descriptor to create two agents of different types.

CDD File for Cache Object Management and the EAR

To set up the project for cache object management, you create a new CDD file using the template provided for cache OM. The CDD template defines an inference agent class and a cache agent class, and default values for cluster level properties, so it doesn't take much to configure an example project to work. Other configuration options are available for different situations.

Learning Points

How do I configure cluster nodes to discover each other?

For the tutorial you can use the default multicast node discovery settings. The default settings are available only when you use Legacy ActiveSpaces as your cluster provider.

You can also change the defaults as needed to avoid collisions with another cluster in the same network. Cluster discovery settings are available for different situations, such as hosts with multiple NICs. Another method of cluster discovery uses well-known addresses for situations where use of multicast is not an option.

How do I configure agents and processing units?

A processing unit deploys as a TIBCO BusinessEvents engine. At a minimum, you list the agents you want to run in one processing unit. For this example, you do not have to do any processing unit configuration because defaults will work out of the box. However, for real-world scenarios, where concurrency features are used, and perhaps add-on products, processing unit configuration requires some thought and planning.

Number of cache agents to start

For a production environment you can define how many cache agents must be started before the system startup can continue. The default is one. When a backing store is used, cache agents can preload objects from a backing store at system startup. More cache agents make the preloading phase quicker. See the Cache Agent Quorum setting in the Object Management Configuration panel.

More Information

See *TIBCO BusinessEvents Configuration Guide*.

Adding and Configuring a CDD

Start with a new CDD because the provided Cache OM template sets many default values you'll need.

Procedure

1. In TIBCO BusinessEvents Studio Explorer, right-click the project name and select **New > Cluster Deployment Descriptor**.

   The New Cluster Configuration Wizard is displayed.

2. In the File Name field, type fdCache and click **Next**.

> ⚠ **Warning:** If you click **Finish** instead of **Next,** you do not have the opportunity to use the Cache OM template. If you accidentally click **Finish**, just delete the file and add a new one, and this time click **Next.**

By default, the CDD file name you specify is also used as the cluster name.

The Object Manager Selection dialog is displayed.

3. Select the following values and click **Finish**.

   - **Cluster Provider Type** - Clustered

   - **Object Management Type** - Cache

The CDD editor is displayed. The default configuration uses ActiveSpaces cluster and cache providers.

This tutorial uses the default cluster and OM but there is an option to change them to FTL cluster and Ignite cache respectively.

4. In the Cluster tab, select **Properties**. You do not need to add any properties, unless you need to change the defaults for the *discovery URL* and the *listen URL* (which is used by the nodes to communicate with each other). Here are the cluster properties with their default values for your information:

   ```
   be.engine.cluster.as.discover.url=tibpgm://7888/;239.8.8.9
   be.engine.cluster.as.listen.url=tcp:// 0.0.0.0  /random port 3000+
   ```

For details see DataGrid Discover URL and DataGrid Listen URL in *TIBCO BusinessEvents Configuration Guide*.

> ℹ **Note:**
> **Cluster Clashes**
>
> If you have clashes with other clusters running on your machine, use a different cluster name, discovery URL and listen URL from the other cluster or clusters.

5. In the Cluster tab, select **Cluster Management** and verify that Legacy ActiveSpaces is selected as the Cluster Provider. In the **Object Management** verify that **Number of Backup Copies** is 1, which indicates that the distributed cache contains one copy of each entity in the cache.

Limited Cache

The **Entity Cache Size** and **Object Table Cache Size** settings in the **Object Management** tab take effect only if you use a limited cache. By default, the cache is not limited. In **Domain Objects > Default** , the setting **Is Cache Limited** is not selected. However when you add a backing store, it is common to use a limited cache, because you can store all objects in the backing store. You can override the limited cache setting at the object level, as desired.

6. As you did in the basic design tutorial, do the following:

- Configure the `inference-class` agent class with the `Channels/Http/AllOps` destination

- Configure the `RuleFunctions/InitializeScorecard` rule function as the startup rule function.

   For details, see Adding and Configuring a CDD.

   No configuration is needed for the cache-agent class.

7. Save and close the CDD.

8. Build the EAR file using the name `fdcache`. If you need a refresher on building the EAR file, see Building the EAR File.

What to do next

You have configured the CDD for cache cluster discovery, and you configured the processing units and agents as you did before. Now you can start the project and see how things work at runtime.

Deployment of Inference and Cache Agents

The agents you have configured are now ready for deployment.

Learning Points

Start cache agents first

In a production environment, you would set the Cache Agent Quorum setting in the Object Management Configuration panel to ensure that enough cache agents start before other types of agents can begin processing. In general, at startup, you start cache

agents then other types of agents. At system shutdown, you stop other kinds of agents, then stop cache agents.

More Information

See Engine Startup and Shutdown Sequence in *TIBCO BusinessEvents Administration*.

Starting the TIBCO BusinessEvents Agents and Sending Events

To start, you can simply open the example `readme.html` file and follow instructions there.

> ✅ **Tip:** In order to send events into the engines, you must enter information into the forms provided in the readme. Navigate to *BE_HOME*/examples/standard/FraudDetectionCache and open `readme.html` in your browser. Follow the instructions in the readme file to start two engines and send events to the inference agent.
>
> See Import of Existing Projects into Your Workspace to understand the requirements for using the interactive readme file.

Procedure

1. Open a command window and at the command prompt, start the TIBCO BusinessEvents engine using the following command.

   This command starts the cache agent. Locations specified are those of the provided configured example. Substitute your values for *BE_HOME*, and the location of the EAR and CDD files (It can be convenient to copy the CDD from your workspace to the same location where you saved the EAR file.):

   ```
   cd BE_HOME/examples/standard/FraudDetectionCache
   BE_HOME/bin/be-engine --propFile BE_HOME\bin\be-engine.tra -u cache
   -c FraudDetectionCache/fdcache.cdd fdcache.ear
   ```

   For example, if you are simply running the provided example, you would use this command line to start the cache agent:

   ```
   c:/tibco/be/6.0/bin/be-engine --propFile c:/tibco/be/6.0/bin/be-
   engine.tra -u cache -c FraudDetectionCache/fdcache.cdd fdcache.ear
   ```

2. When the cache agent has started, open a second command window, and start the

inference agent in a similar way. Again, locations specified are those of the provided pre-configured example.

```
cd BE_HOME/examples/standard/FraudDetectionCache
BE_HOME/bin/be-engine --propFile BE_HOME\bin\be-engine.tra -u cache
-c FraudDetectionCache/fdcache.cdd fdcache.ear
```

For example, if you are simply running the provided example, you would use this command line to start the cache agent:

```
c:/tibco/be/6.0/bin/be-engine --propFile c:/tibco/be/6.0/bin/be-
engine.tra -u cache -c FraudDetectionCache/fdcache.cdd fdcache.ear
```

```
cd BE_HOME/examples/standard/FraudDetectionCache
BE_HOME/bin/be-engine --propFile BE_HOME\bin\be-engine.tra -u
default -c FraudDetectionCache/fdcache.cdd fdcache.ear
```

For example, if you are simply running the provided example, you would use this command line:

```
c:/tibco/be/6.0/bin/be-engine --propFile c:/tibco/be/6.0/bin/be-
engine.tra -u default -c FraudDetectionCache/fdcache.cdd
fdcache.ear
```

3. Optional Activity

   Start a second cache agent. Experiment with stopping one cache agent engine then restarting it again, to see how one cache agent can operate successfully. Because one backup copy of the data is kept, and there are two cache agents, each cache agent has the complete cache contents. In a production system with more cache agents no single cache agent has all the cache data.

Result

Congratulations — You have completed the caching tutorial!

You can now continue to Backing Store Tutorial.

Backing Store Tutorial

For this tutorial, you can use either the project you have developed in the previous two tutorials, or the provided example project.

After you have prepared a project described in Cache OM Tutorial, you are ready to add a backing store to it.

To complete just the backing store tutorial, you can begin with the provided cache object management tutorial project:

BE_HOME/examples/standard/FraudDetectionCache

Finally, if you do not want to complete the steps in the backing store tutorial, you can refer to the fully configured example project, provided in

BE_HOME/examples/standard/FraudDetectionStore.

This tutorial guides you through the basic steps required to set up a backing store for the Fraud Detection Cache project. The "Backing Store Setup" section in *TIBCO BusinessEvents Developer's Guide* covers more cases. The tutorial organizes the steps to create a convenient flow, as follows:

First, Prerequisites: DBMS Product Setup

The tutorial explains the prerequisites in general. Refer to the *TIBCO BusinessEvents Readme* for DBMS product requirements.

Verifying DBMS Software and Driver and Editing the TRA.

Then, TIBCO BusinessEvents Project Configuration

Some TIBCO BusinessEvents Studio project configuration can be done before or after database schema setup. If ontology-related configuration is required, however, it must be completed before running the database setup scripts.

Ontology-related configuration may be required for special cases that you won't deal with in this example project. (See the section Cases That May Need Additional Setup in TIBCO BusinessEvents Developer's Guide for more details.) Ontology-related configuration is also required if you want to exclude any entity types from the backing store (or to include types that were earlier excluded). Cache preloading settings are not related to backing store setup, and can be configured and changed at any time.

- Renaming the Fraud Detection Cache Project and Changing Port.

- Adding a JDBC Connection Shared Resource.

- Renaming and Configuring the CDD and Building the EAR.

Finally, Database Schema Setup

Schema setup is facilitated by the scripts provided with the product. The tutorial uses default values throughout. The scripts can also be used to reset the project.

- Running the Initialize Database Script as the DBA or System User.

- Running the Create Tables Scripts as the TIBCO BusinessEvents User.

- Generating the Project-Specific SQL Scripts.

- Running the Table Creation Script

Verifying DBMS Software and Driver and Editing the TRA

If you are using Oracle Database, you do not have to edit the TRA file for this tutorial.

Before you begin to configure the backing store, do the following:

Procedure

1. Install and start a supported DBMS product. See the product readme file for a list of supported products. This tutorial uses Oracle Database 12c Enterprise Edition. If you use a different supported database product, adapt the instructions accordingly.

2. Copy the appropriate JDBC drivers file (for example, `ojdbc7.jar`) to *BE_HOME*`/lib/ext/tpcl`.

   You can download Oracle JDBC drivers from Oracle.com.

3. You must restart BusinessEvents Studio Explorer after copying the drivers file. This step is required before you can use the design-time Test Connection feature. It is also required for runtime.

4. If you will use the debugger or tester features, add your DBMS product's libraries to the TIBCO BusinessEvents Studio classpath. See the section Working with External Library and Custom Function Paths in TIBCO BusinessEvents Developer's Guide.

Result

You have taken care of basic DBMS prerequisite steps. Next step is to configure the TIBCO BusinessEvents Studio project.

TIBCO BusinessEvents Studio Project

Except for excluding entities from the backing store, modifying of the FraudDetectionCache project to support a backing store can be configured before or after setting up the backing store, and can be changed after the backing store is in place.

Can I exclude entities from the backing store?

Yes. You can configure various settings at the individual entity type level. To exclude entities, use the **CDD Cluster tab > Object Management > Domain Objects > Overrides** feature. Clear an entity override's **Has Backing Store** checkbox to exclude it from the backing store. See "Excluding Entities from the Backing Store" in *TIBCO BusinessEvents Developer's Guide* for more details (including another way to exclude entities, using their Mode setting).

> ℹ **Note:** If later you want to include any excluded entities, you must change the setting and update the backing store setup as explained in "Updating an Existing Backing Store Schema" in *TIBCO BusinessEvents Developer's Guide*

What is preloading?

When you use a backing store, you can preload entity data from the backing store to the cache before the system begins processing events. If you do not preload entities, they are fetched from the backing store as required at runtime (which means the first time objects are fetched is slower).

Why preload handles?

Entity object handles are stored in a special cache that can be preloaded or not depending on storage and performance needs. It might be more efficient in some cases to preload handles and not objects, for example. See The Role of the Object Table in TIBCO BusinessEvents Architect's Guide for more details.

Preloading commonly used objects can improve performance after startup. By default, no preloading is done. You can preload all objects or objects of selected entity types, as

desired. See Domain Objects Configuration in TIBCO BusinessEvents Developer's Guide for the procedure.

More Information

In *TIBCO BusinessEvents Developer's Guide*, see Backing Store Configuration, and JDBC Connection Reference.

Renaming the Fraud Detection Cache Project and Changing Port

You need to modify the Fraud Detection Cache project to support a backing store. To configure the backing store behavior at runtime, use settings that are available.

Procedure

1. Open TIBCO BusinessEvents Studio and, as needed, import the FraudDetectionCache project into your workspace. See Import of Existing Projects into Your Workspace.

2. In TIBCO BusinessEvents Studio Explorer, right-click the project name and select **Refactor > Rename**. In the **New Name** field, type **FraudDetectionStore** and click **Preview**. You can preview the effect of this change, then click **OK**.

3. Open the **HTTPConnection** resource that's in the **SharedResources** folder and change the port to **8209**.

   The provided FraudDetectionStore example project ships with this port, and its `readme.html` points to it as well (see Dependency between Readme File and Project — Port Number for more details).

4. Save the changes and close the resource.

Adding a JDBC Connection Shared Resource

In this task, you add a JDBC Connection shared resource to your project and configure it to connect to the backing store database. You can do this before creating the database if the necessary details are known.)

The following details assume you will connect to a local instance of Oracle Database 12c Enterprise Edition. Adapt the instructions as needed for your database product.

> **ℹ** **Note:** You will specify the location of this JDBC Connection resource in the CDD, in Renaming and Configuring the CDD and Building the EAR .

Procedure

1. In TIBCO BusinessEvents Studio, open your project (if it is not already open), right-click the **SharedResources** folder and select **New > Other > TIBCO Shared Resources > JDBC Connection** (or select the folder and press Ctrl-N as in earlier tutorials).

2. At the New JDBC Connection Wizard dialog, name the connection **ForBackingStore**, and click **Finish**.

    You see the JDBC Connection resource editor.

    > **✓** **Tip:** The globe icons to the right of the settings in this resource type indicate that you can use global variables in the field value. To learn more about this topic, see Working with Global Variables in *TIBCO BusinessEvents Developer's Guide*.

3. Optionally, enter a description such as `JDBC Connection for the backing store tutorial`.

4. In the JDBC Driver field, select the appropriate driver for your database. The tutorial example uses **oracle.jdbc.OracleDriver**. The database URL format appears in the Database URL field just below:

    ```
    jdbc:oracle:thin:@<host>:<port#>:<db_instancename>
    ```

5. In the Database URL field, configure the URL with your actual values. The tutorial uses these values:

    ```
    jdbc:oracle:thin:@localhost:1521:XE
    ```

    Where `1521` is the default port, and `XE` is the default instance name for Oracle Database 12c Enterprise Edition. (The default instance name for Oracle Database 12c is `ORCL`.)

6. In the User Name and Password fields, type the name `BE_USER` and password `BE_USER`. (The password field does not display the text.). These are the username and

password of the database user you will create (in Running the Initialize Database Script as the DBA or System User).

7. Save and close the resource.

Renaming and Configuring the CDD and Building the EAR

Begin by renaming the CDD from the Cache OM Tutorial, and add a few configuration details for backing store functionality: click a checkbox to enable backing store functionality, and select the JDBC Connection resource you want to use. Of course, more options are available for different needs.

Procedure

1. In TIBCO BusinessEvents Studio Explorer, right-click the `fdcache.cdd` resource and select **Refactor > Rename**. In the New Name field, type `fdstore` and click **Preview**. You can preview the effect of this change, then click **OK**.

2. Open the newly renamed `fdstore` CDD file in the resource editor.

3. Click the **Cluster** tab and select the **General** section on the left. Change the cluster name from `fdcache` to `fdstore`.

   > **Note:**
   > **Cluster Clashes**
   >
   > If you have clashes with other clusters running on your machine, use a different cluster name, discovery URL and listen URL from the other cluster or clusters. See Cache OM Tutorial > Adding and Configuring a CDD for details.

4. Select **Cluster tab > Object Management > Persistence**

   a. Select Persistence Option **Store**.

   b. Select Store Type **Oracle** (or as needed for your DBMS).

5. Select **Cluster > Object Management > Persistence > Connection** on the left and in the URI field, select the JDBC Connection shared resource you configured for the backing store (see Adding a JDBC Connection Shared Resource).

   You can optionally check that the rest of the CDD configuration is correct. Refer to

Adding and Configuring a CDD.

6. Select **Domain Objects > Default** on the left.

   The Domain Objects area is where you configure how objects are managed. You can set defaults, and you can override them for specific entities.

   - On the right, ensure that Mode is set to `Cache Only` (the default for the Cache OM CDD template).

   - Check the **Preload Entities** and **Preload Handles** checkboxes.

     Preloading is not necessary for the tutorial to function, but it is a commonly used feature. See section introduction for more details.

7. If there are override entries, check each entry in turn, and make sure that the Mode is set to `Cache Only`, and that the **Has Backing Store** checkbox is selected.

8. Save the CDD file.

9. In TIBCO BusinessEvents Studio Explorer, highlight the project name, then from the top menus select **Project > Build Enterprise Archive**.

   If you see a message asking you to save all project resources, click **Yes**. It means an unsaved resource editor is open.

   a. Change the Name to `FraudDetectionStore`

   b. In the **File Location** setting, specify a location from which you can access the EAR when you run the database setup utility and when you start the engine.

   c. At the end of the location string, change the file name to `fdstore.ear`. The FraudDetectionStore example ships with this value: *BE_HOME*`/examples/standard/FraudDetectionStore/fdstore.ear`.

     If you need a refresher on building the EAR file, see Building the EAR File.

What to do next

Now you are ready to run the scripts that prepare the backing store's database schema.

Database Schema

Use provided scripts to set up the database schema.

> ✓ **Tip:** See Resetting the Backing Store Tutorial for various levels of reset you can do if you want to reuse this tutorial example.

What happens if I change the project ontology after setting up the database schema?

If you change the project ontology you must update the database schema. You must also update the schema if you add an entity that you previously excluded from the backing store.

A utility is provided that can handle most changes. Certain changes require manual updates. See the section "Updating an Existing Backing Store Schema", in *TIBCO BusinessEvents Developer's Guide*.

What are the minimum permissions for the database user?

At a minimum, the user must be able to create tables and views. By default the `BE_USER` user created by the scripts has DBA privileges.

Running the Initialize Database Script as the DBA or System User

This script creates the TIBCO BusinessEvents user and initializes the database.

The tutorial uses the default username (`BE_USER`) and password (`BE_USER`). You can change the username by editing the *BE_HOME*/bin/ `initialize_database_YourDBMS`.sql script.

> ⚠ **Warning:** Running the `initialize_database_YourDBMS`.sql script deletes the user before creating it again. Running the `create_tables_YourDBMS`.sql drops all database tables before creating them again. This means you can run these scripts again during test phases of your project development, without having to take extra cleanup steps.

Procedure

1. Ensure your DBMS is running. Open a command window in *BE_HOME*/bin (default location of the scripts) and type the following command for Oracle (if you use the default SID you can omit *@SID*):

```
sqlplus sys_user/sys_user_password@SID/@initialize_database_oracle.sql
```

Type exit to exit and commit.

If you are using SQL Server, use this instead:

```
osql -S Your-Server -U sys_user -P sys_user_password -n -i initialize_database_
sqlserver.sql
```

This script creates the TIBCO BusinessEvents database user. This user must be used to run the other scripts. You see messages like the following:

```
    User dropped.
    User created.
    Grant succeeded.
```

> ✅ **Tip:** Using your database product, you can configure additional users to access the database, in addition to this user.

Running the Create Tables Scripts as the TIBCO BusinessEvents User

Next you log on as the TIBCO BusinessEvents user, BE_USER by default, and run a script to create non-project-specific tables.

Procedure

1. Still in the command window in BE_HOME/bin, type the following command for Oracle.

```
sqlplus BE_USER/BE_USER@SID @create_tables_oracle.sql
```

Type exit to exit and commit.

If you are using SQL Server, use this instead:

```
osql -S Your-Server -d Your-DB -U BE_USER -P BE_USER -n -i @create_
tables_sqlserver.sql
```

Result

You see various harmless error messages the first time you run the script, and various messages saying `Table created` and `Index created`.

Generating the Project-Specific SQL Scripts

Ensure that you have generated the project EAR file before beginning.

Procedure

1. In TIBCO BusinessEvents Studio Explorer, right-click the `FraudDetectionStore` project name and select **Export > TIBCO BusinessEvents > Backingstore Deployment**.

2. Click **Next**.

   You will see the Deploy wizard.

3. In the **Backingstore Type** field, select `Relational Database (JDBC)` and click **Next**.

4. Complete the values as follows.

   a. In the **Database Type** field, select your database type. The default is `oracle`.

   b. In the **Cluster Deployment Descriptor** field, browse to and select the `fdstore` CDD.

   c. In the **Output Directory** field, browse to and select the directory where the scripts will be generated. The tutorial project uses BE_HOME/`bin`.

   d. In the **Output Script Name Prefix** field, type `fdstore`.

5. Click **Finish**.

   The generated scripts appear in the directory you specified. If you are following the tutorial instructions, the following files appear in the BE_HOME/`bin` directory:

   ```
   fdstore.sql
   fdstore_cleanup.sql
   fdstore_delete.sql
   fdstore_alter.sql
   fdstore_remove.sql
   fdstore.aliases
   ```

   The use of these scripts is explained in "Resources Required for JDBC Backing Store

Implementation" in *TIBCO BusinessEvents Developer's Guide*.

Running the Table Creation Script

Now you can run the table creation script.

Procedure

1. Open a command window in *BE_HOME*/`bin` and type the following command.

```
sqlplus BE_USER/BE_USER @fdstore.sql
```

> ℹ️ **Note:** For SQL Server:
>
> ```
> osql -S Your-Server -d Your-DB -U BE_USER -P BE_USER -n -i
> @create_tables_oracle.sql
> ```

Result

Congratulations! You are ready to start the `FraudDetectionStore` application.

Application Deployment and Testing

After restarting the system, data persists. With a cache-based system, you lose all data in the event of a total system shutdown. With the backing store in place, data remains available after the system restarts. You can experiment with this by adding data then restarting all engines.

More Information

See "Engine Startup and Shutdown Sequence" in *TIBCO BusinessEvents Administration*.

Starting the TIBCO BusinessEvents Agents

You can simply open the example `readme.html` file and follow instructions there

See Dependency of Interactive Readme File on a Higher Level Directory to understand the requirements for using the interactive readme file.

> ✓ **Tip:** In order to send events into the engines, you must enter information into the forms provided in the readme. Navigate to *BE_HOME*/examples/standard/FraudDetectionStore (or the location where you copied it to) and open readme.html in your browser. Follow the instructions in the readme file to start two or more engines and send events to the inference agent.

Procedure

1. Open a command window and, at the command prompt, start the TIBCO BusinessEvents engine using the following command.

   ```
   cd BE_HOME/examples/standard/FraudDetectionStore
   BE_HOME/bin/be-engine --propFile
   BE_HOME/bin/be-engine.tra -u cache -c
   FraudDetectionStore/fdstore.cdd fdstore.ear
   ```

   You can substitute your values for *BE_HOME*, and the location of the EAR and CDD files.

   For example, if you are simply running the provided example, you would use this command line to start the cache agent:

   ```
   c:/tibco/be/6.0/bin/be-engine --propFile c:/tibco/be/6.0/bin/be-
   engine.tra -u cache -c FraudDetectionStore/fdstore.cdd fdstore.ear
   ```

   This command starts the cache agent. Locations specified are those of the provided configured example.

2. When the cache agent has started, open a second command window, and start the inference agent in a similar way. Again, locations specified are those of the provided preconfigured example.

   ```
   cd BE_HOME/examples/standard/FraudDetectionStore
   BE_HOME/bin/be-engine --propFile
   BE_HOME/bin/be-engine.tra -u default -c
   FraudDetectionstore/fdstore.cdd fdstore.ear
   ```

   For example, if you are simply running the provided example, you would use this command line to start the inference agent:

```
c:/tibco/be/6.0/bin/be-engine --propFile c:/tibco/be/6.0/bin/be-
engine.tra -u default -c FraudDetectionStore/fdstore.cdd
fdstore.ear
```

Sending Events

Send some events, such as that one more debit could cause an account to be suspended.

Stop both engines. Then start them up again. Send one or more events such that the threshold for account suspension is crossed. You'll see that the account is suspended, based on data that was retrieved from the backing store as well as data you just entered.

Resetting the Data

See Resetting the Backing Store Tutorial.

Congratulations — You have completed the backing store tutorial!

Resetting the Backing Store Tutorial

You can reset the tutorial for re-use.

Different levels of reset are available as explained next:

To do this:	Do the following:
Remove the backing store data only.	Run `FDStore_cleanup.sql`. This script truncates the tables.
Remove the database schema and reset the project.	Run `FDStore_remove.sql`. To recreate a fresh schema, do Running the Create Tables Scripts as the TIBCO BusinessEvents User and Generating the Project-Specific SQL Scripts .
Remove the database user and the database schema.	Do all tasks in the section Database Schema. The `initialize_database_dbms.sql` script drops the user (and therefore all the tables) and then adds the user again.

TIBCO Documentation and Support Services

For information about this product, you can read the documentation, contact TIBCO Support, and join TIBCO Community.

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the TIBCO Product Documentation website, mainly in HTML and PDF formats.

The TIBCO Product Documentation website is updated frequently and is more current than any other documentation included with the product.

Product-Specific Documentation

Documentation for TIBCO BusinessEvents® Enterprise Edition is available on the TIBCO BusinessEvents® Enterprise Edition Product Documentation page.

To directly access documentation for this product, double-click the file at the following location:

TIBCO_HOME/release_notes/TIB_businessevents-enterprise_6.1.2_docinfo.html

where *TIBCO_HOME* is the top-level directory in which TIBCO products are installed. On Windows, the default *TIBCO_HOME* is C:\tibco. On UNIX systems, the default *TIBCO_HOME* is /opt/tibco.

The following documentation for this product is available on the TIBCO BusinessEvents® Enterprise Edition Product Documentation page:

- *TIBCO BusinessEvents® Release Notes*
- *TIBCO BusinessEvents® Installation*
- *TIBCO BusinessEvents® Getting Started*
- *TIBCO BusinessEvents® Architect's Guide*
- *TIBCO BusinessEvents® Administration*
- *TIBCO BusinessEvents® Developer's Guide*
- *TIBCO BusinessEvents® Configuration Guide*
- *TIBCO BusinessEvents® Data Modeling Developer's Guide*

- *TIBCO BusinessEvents® Decision Manager User's Guide*

- *TIBCO BusinessEvents® WebStudio User's Guide*

- *TIBCO BusinessEvents® Event Stream Processing Pattern Matcher Developer's Guide*

- *TIBCO BusinessEvents® Event Stream Processing Query Developer's Guide*

- *TIBCO BusinessEvents® Security Guide*

- Online References:

  - *TIBCO BusinessEvents® Java API Reference*

  - *TIBCO BusinessEvents® Functions Reference*

Other TIBCO Product Documentation

When working with TIBCO BusinessEvents Enterprise Edition, you may find it useful to read the documentation of the following TIBCO products:

- TIBCO ActiveSpaces®: It is used as the cluster, cache, or store provider for the TIBCO BusinessEvents Enterprise Edition project.

- TIBCO FTL®: It is used as the cluster provider for the TIBCO BusinessEvents Enterprise Edition project.

- TIBCO Streaming®: It is used as the metrics store provider for the TIBCO BusinessEvents Enterprise Edition project.

How to Access Related Third-Party Documentation

When working with TIBCO BusinessEvents® Enterprise Edition, you may find it useful to read the documentation of the following third-party products:

- Apache Ignite

- InfluxDB

- Grafana

- Apache Cassandra

How to Contact TIBCO Support

Get an overview of TIBCO Support. You can contact TIBCO Support in the following ways:

- For accessing the Support Knowledge Base and getting personalized content about

products you are interested in, visit the TIBCO Support website.

- For creating a Support case, you must have a valid maintenance or support contract with TIBCO. You also need a user name and password to log in to TIBCO Support website. If you do not have a user name, you can request one by clicking **Register** on the website.

How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the TIBCO Ideas Portal. For a free registration, go to TIBCO Community.

Legal and Third-Party Notices

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE "LICENSE" FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIBCO, the TIBCO logo, the TIBCO O logo, ActiveMatrix, ActiveMatrix BusinessWorks, ActiveSpaces, TIBCO Administrator, TIBCO BusinessEvents, TIBCO Designer, Enterprise Message Service, TERR, TIBCO FTL, Hawk, TIBCO LiveView, TIBCO Runtime Agent, Rendezvous, Statistica, and StreamBase are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

Java and all Java based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

This document includes fonts that are licensed under the SIL Open Font License, Version 1.1, which is available at: https://scripts.sil.org/OFL

Copyright (c) Paul D. Hunt, with Reserved Font Name Source Sans Pro and Source Code Pro.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

This software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the readme file for the availability of this software version on a specific operating system platform.