

IBM Operational Decision Manager
Version 8 Release 5

Getting Started with Event Rules



Note

Before using this information and the product it supports, read the information in "Notices" on page 99.

This edition applies to version 8, release 5, modification 1 of Operational Decision Manager and to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright IBM Corporation 2008, 2013.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Tutorials 1

Getting started with event rules	1
Exercise 1: Building a simple application	3
Task a: Creating an event definition and an action definition	4
Task b: Defining connections to the systems	10
Task c: Building the business logic	13
Task d: Deploying the event project	15
Task e: Testing the event project	16
Exercise 2: Adding conditional logic to the application	18
Task a: Writing a condition in the event rule	19
Task b: Verbalizing a business object field to define a context relationship	20
Task c: Testing the context relationship between events	23
Exercise 3: Defining another event and building another condition	25
Task a: Define the policy purchased event	25
Task b: Create the filter	27
Task c: Add the filter to the event rule	28
Task d: Create the event rule	28
Task e: Deploy and test the event project and the event rule.	29
Exercise 4: Creating event rules with delays	30
Task a: Define the action	31
Task b: Create the event rule.	32
Task c: Create another condition in the event rule	33
Task d: Update the event rule	34
Task e: Deploy and test the event project and the event rule.	35
Tutorial: Calculating averages to identify event patterns	37
Before you start	37
Tutorial scenario.	38
Running the completed tutorial.	38
Task 1: Creating an event project	40
Task 2: Defining the business objects	42

Task 3: Defining the event rule and filters	47
Task 4: Deploying the application	50
Tutorial: Comparing an event with a previous event	50
Before you start	50
Tutorial scenario.	51
Running the completed tutorial.	52
Task 1: Creating an event project	53
Task 2: Defining the business objects	55
Task 3: Defining the event rule and filter	60
Task 4: Deploying the application	62
Tutorial: Tracking the state of something.	63
Before you start	63
Tutorial scenario.	64
Viewing the completed tutorial.	64
Task 1: Creating an event project	66
Task 2: Defining the business objects	70
Task 3: Defining the event rules and filter	75
Task 4: Deploying the application	77
Tutorial: Maintaining a running total	78
Before you start	78
Tutorial scenario.	79
Running the completed tutorial.	79
Task 1: Creating an event project	80
Task 2: Defining the business objects	83
Task 3: Defining the event rule and filter	88
Task 4: Deploying the application	90
Tutorial: Invoking a ruleset from an event project.	90
Before you start	91
Tutorial scenario.	92
Running the completed tutorial.	92
Task 1: Defining a Business Rules Data Connection	94
Task 2: Creating an event rule to use the data connection.	96
Task 3: Deploying the tutorial application	97

Notices 99

Trademarks	101
----------------------	-----

Tutorials

Develop your technical skills by completing the tutorials for Decision Server Events.

Getting started with event rules

In this tutorial, you build a small application around the scenario of an insurance company that is identifying business opportunities by using business event processing.

Scenario

An insurance company wants to improve its ability to identify potential customers and close sales that might otherwise be lost. The company sells a range of insurance products, but this tutorial focuses on identifying potential car insurance customers. The insurance company has a strong online presence. Most of the requests for quotations that come directly from potential customers (rather than through brokers or agents), come through the company website.

When an individual requests a quotation for car insurance using the company website, the request comes to the company quotation system. The quotation system responds by displaying a quotation on the website. A follow-up email is also sent to confirm the quotation, which is then valid for up to 90 days (12 weeks).

When the quotation request arrives via the website, it is also sent to Decision Server Events. A large number of quotation requests are received every day. It is too expensive to follow up every quotation with a telephone call. The company want to identify when a quotation request is most likely to lead to a sale. In this situation the potential customer receives a personal call from the company call center.

For example, the company believe that a potential customer shows that they are more seriously looking to buy a car insurance policy when they request more than one quotation for car insurance, from the company in a short time period (for example, altering the options selected on the policy). The insurance company wants to add the potential customer to a relevant marketing campaign.

Because the insurance company does not want to annoy customers, it is important that the potential customer does not receive a sales call from the call center if they have already purchased a policy from the company. It is also important that the customer is not added to marketing campaigns multiple times, if they later request a second or third quotation from the company.

Before you start the tutorial

1. Install Decision Server Events by following either the product launchpad install procedure, or the IBM® Installation Manager install procedure:
 - Install the product and the sample server by using the product launchpad. See Installing the product and the sample server. This creates a sample server profile on WebSphere® Application Server to run the samples and tutorials, and installs the required components to perform this tutorial.

- Install the product by using the IBM Installation Manager and then selecting the Event Tester Enabler feature during the installation. See Selecting the features to install. The Event Tester Enabler feature is not selected by default during the installation, but is required to perform this tutorial. By default, the IBM Installation Manager installs Event Designer, the event runtime, and the Event Widgets for Decision Server.

Important: If you previously installed Decision Server Events and there is an existing sample server profile on your system, you must remove the older profile and create a new profile to enable the Event Tester. See Restoring the sample server.

2. Click `Getting_Started_with_Events.zip` to download this zip file to your computer and extract it to a directory on your computer. The `Getting_Started_with_Events.zip` file contains a completed version of the tutorial application.

You can now work through the tutorial exercises.

Tutorial exercises

The first exercise in this tutorial focuses on building an application that adds potential customers to marketing campaigns, when they request a quotation for car insurance. Subsequent exercises increase the capability of the application by adding further logic to identify when a customer has requested more than one quotation for car insurance in a specific time period.

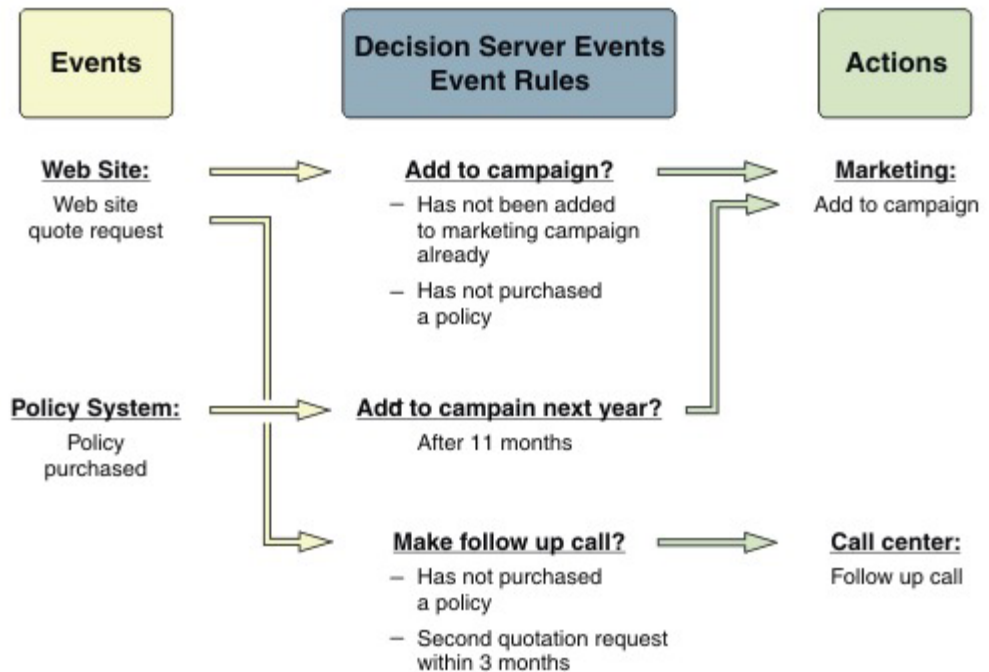
Each exercise introduces new features of Decision Server Events as they become relevant to the application you are building.

The tasks in the insurance tutorial are identified by the type of user who typically completes them in a real-life application:

- IT user - defines the data model (the structure of the events and actions) and the connections to other systems
- Business user - defines the business logic by using the assets that the IT user has previously defined

During the lifetime of the application, if the business logic is to be extended, the IT user might first have to extend the data model, which is illustrated in subsequent exercises of the tutorial.

The following diagram illustrates the scenario used in the tutorial.



Exercise 1: Building a simple application

In this exercise, you create a simplified version of the final application, starting with the basic building blocks. This application is the foundation of the following exercises.

Before you begin

Complete the steps in “Getting started with event rules” on page 1.

About this task

Each time a potential customer requests a quotation for car insurance on the insurance company website, the website sends a business event to the event runtime server. The business event contains the information that the potential customer provided on the website. The website always structures the content of the event in a specific way so that the event runtime server can recognize it. The structure of that event is defined by the IT user in Event Designer, and is stored in Decision Server Events so that incoming events are always recognized.

When it is appropriate to add the potential customer to a marketing campaign, the event runtime sends an action to the Marketing department. The structure of the action is also defined in Event Designer by the IT user. The information in the action might be the same or different from the information in the event to which it is responding.

In this scenario, the potential customer has supplied their name, contact details, and the details of the car they want to insure. The same information is sent to the Marketing department so that the Marketing department can send relevant marketing material to the customer, using the supplied contact details.

Follow steps a-e to complete exercise 1.

Task a: Creating an event definition and an action definition

About this task

You define an event called Website Quote Request to represent the quotation request received from the website. You also define an action called Add To Campaign that contains the information from the event that is required by the Marketing department to add this potential customer to relevant marketing campaigns.

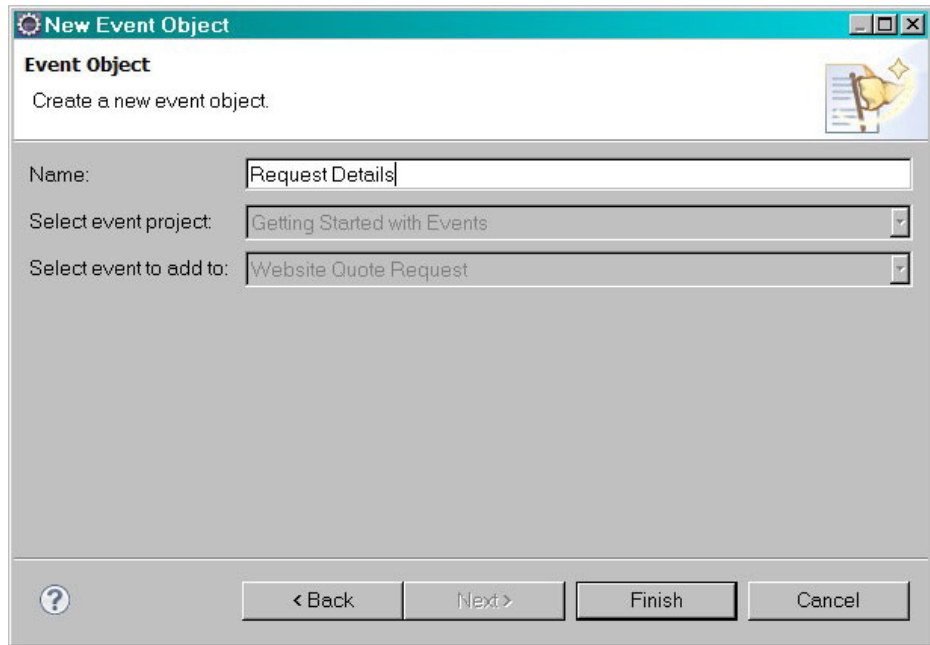
Procedure

1. Start Event Designer:
 - On Windows, click **Start > All Programs > IBM > Operational Decision Manager V8.5.1 > Event Designer**.
 - On Linux, click **Applications > IBM > Operational Decision Manager V8.5.1 > IBM Decision Server V8.5.1 > Event Designer**.
2. Create an event project to contain the tutorial application:
 - a. In the Event Explorer view, right-click anywhere and then click **New > Event Project**.
 - b. Name the project Getting Started with Events and click **Finish**.
3. Define the Website Quote Request event:
 - a. In the Event Explorer view, right-click Getting Started with Events, then click **New > Event**.
 - b. In the New Event wizard, select **Create a blank event** and click **Next**.
 - c. Name the event Website Quote Request and click **Finish**. The wizard closes and an editor for the new event is opened.

You have defined that the business event received from the website contains information that is held within the structure of the Website Quote Request event.

Now, you must define what that structure is, by defining an event object for the event.
 - d. In the Website Quote Request editor, there is an expanded section called Event Objects. In the Event Objects section, click **Add** and the New Event Object wizard is displayed.
 - e. Select **Add a new blank event object** and click **Next**.
 - f. Name the event object Request Details and click **Finish**.

The following screen capture shows the New Event Object wizard:



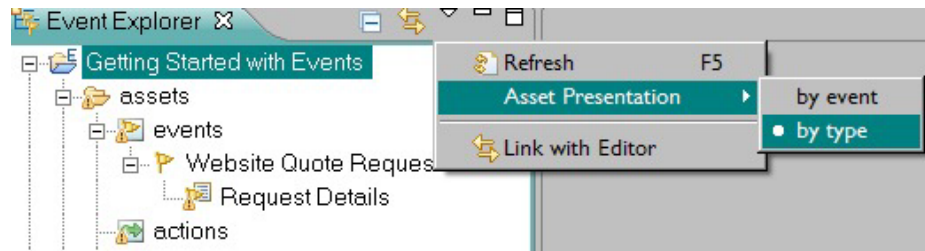
The Request Details event object contains fields that represent the discrete pieces of information about the customer and their car in the Website Quote Request event. For example, the name of the customer and the year the car was first registered. Event objects are separate assets, even though they represent parts of an event, which enables you to optionally reuse a single event object in multiple events.

- g. Press Ctrl+S to save the Website Quote Request editor updates. Close the Request Details editor tab.

The following screen capture shows the Getting Started with Events project in the Event Explorer view. You can also see that the Event Explorer **Asset Presentation** view is **by type**. To change the asset presentation in the Event Explorer view, click the View Menu icon which is the down arrow (

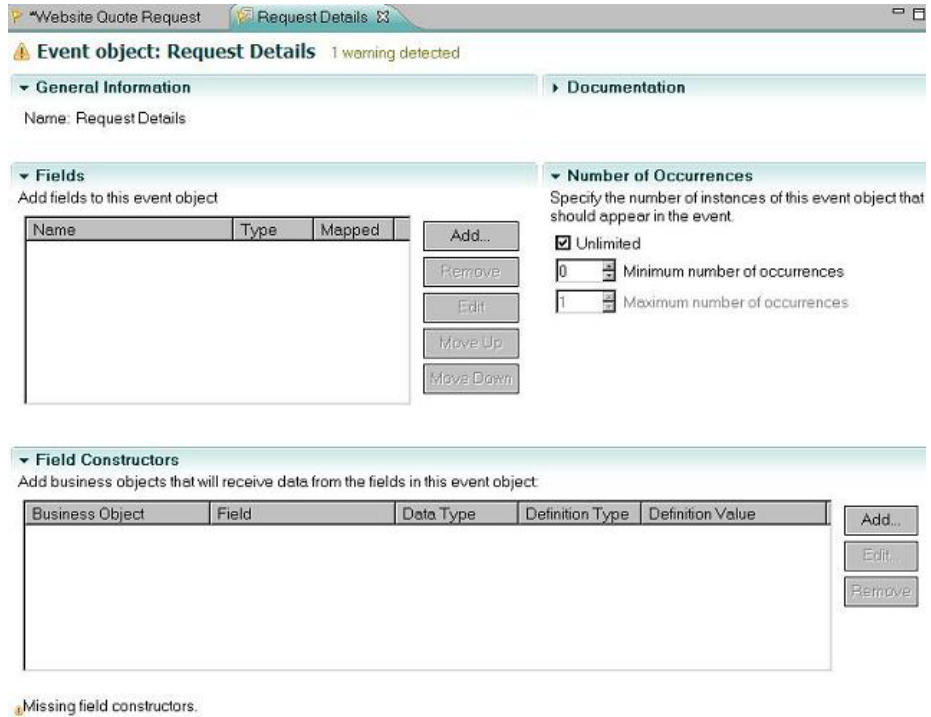


) to the right of the **Event Explorer** tab.



- h. To add a field to the Request Details event object in the Website Quote Request editor, double-click the Request Details event object. The Request Details event object opens in a separate editor.

The following screen capture shows the Request Details event object:



- i. Within the Request Details editor, in the section called **Fields**, click **Add** to open the New field wizard.
 - In the Field name, type First Name.
 - For the Data type, click **String**.
 - In Documentation, type First name of customer and click **Finish**.

The field that you just created is displayed.

- j. Add the other fields listed in the following table in the same way. Then close both of the editor tabs and save your changes.

Table 1. Request Details fields to be added.

Name	Data type	Description
First Name	String	First name of customer
Last Name	String	Last name of customer
Zip Code	String	Zip code of customer
Phone	String	Phone number of customer
Registration	String	Car registration number
Year	Integer	Year of first registration of the car

The following screen capture shows the Request Details event object with the fields, that you have just added, listed in the **Fields** table. The Mapped column displays False because the event object fields are not mapped to any business object fields, by using field constructors. You create the field constructors later in this exercise.

⚠ **Event object: Request Details** 1 warning detected

▼ **General Information**

Name: Request Details

▼ **Fields**

Add fields to this event object

Name	Type	Mapped	
• First Name	String	False	<input type="button" value="Add..."/> <input type="button" value="Remove"/> <input type="button" value="Edit"/> <input type="button" value="Move Up"/> <input type="button" value="Move Down"/>
• Last Name	String	False	
• Zip Code	String	False	
• Phone	String	False	
• Registration	String	False	
• Year	Integer	False	

k. Press Ctrl+S to save your updates and close the editor tabs.

You have defined an event called Website Quote Request that contains six fields.

Next, you define the Add To Campaign action.

The Add To Campaign action is sent to the Marketing department every time the event runtime receives a Website Quote Request event. The Add To Campaign action therefore must contain the contact details for the potential customer, and information about the car that is to be insured so that only relevant marketing material is sent to the potential customer.

You define actions in a similar way to defining events.

4. Define the Add To Campaign action:
 - a. In the Event Explorer view, right-click Getting Started with Events, then click **New > Action**.
 - b. In the New Action wizard, select **Create a blank action** and click **Next**.
 - c. In the New Action wizard, enter the action name Add To Campaign, then click **Finish**.
 - d. Press Ctrl+S to save your updates and close the editor tab.

In the Event Explorer view, you can see only the new Add To Campaign action if you are using the **by type** asset presentation. If you are using the **by event** asset presentation, the action is only displayed if it is included in an event rule to be processed by events.

For now, the Add To Campaign action is going to contain the same information as the Website Quote Request event. The Marketing department can use this information to decide which marketing campaigns are relevant to the potential customer. In the next step, you define that the information in the Website Quote Request event is copied into the Add To Campaign action, to be sent to the Marketing department.

5. Define that the information in the Website Quote Request event is used in the Add To Campaign action:

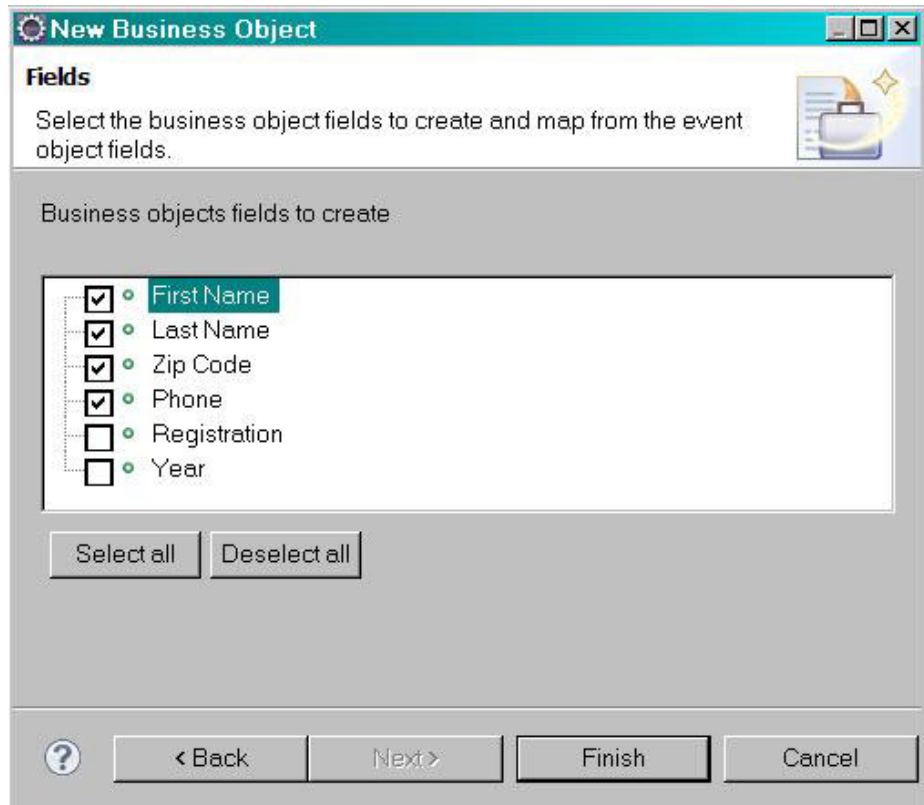
It is actually the information in the Request Details event object that is copied to the Add To Campaign action, where the information is structured in two action objects called Customer and Car.

Between events (incoming to the event runtime) and actions (outgoing from the event runtime) are business objects. Business objects provide abstract representations of the information, or data, that is received in the event, and which can be obtained from elsewhere or through calculations. In this exercise, you create two business objects which contain details of how the information in the Website Quote Request event is copied to the Add To Campaign action.

You create two business objects, Customer and Car, which receive information from the Website Quote Request event. The Customer business object contains the details of the customer, while the Car business object contains the details of the car. Structuring the data by using multiple business objects is a logical way to model the data and it improves the management of the data. For example, if you were to extend the scenario of this tutorial to include the quotation process itself, you might use the Car business object again as part of an insurance policy intermediate object.

- a. In the Event Explorer view, right-click the Request Details event object, then click **Create Business Object from this Event Object**.
- b. In the New Business Object wizard, type Customer in the Name field, and click **Next**.
- c. Select the following business object fields:
 - First Name
 - Last Name
 - Zip Code
 - Phone

The following screen capture shows the selected fields in the New Business Object wizard.



Click **Finish**.

- d. Right-click the Request Details event object again. Click **Create Business Object from this Event Object** and name the new business object Car. Click **Next**.

Select the following business object fields:

- Registration
- Year

Click **Finish**.

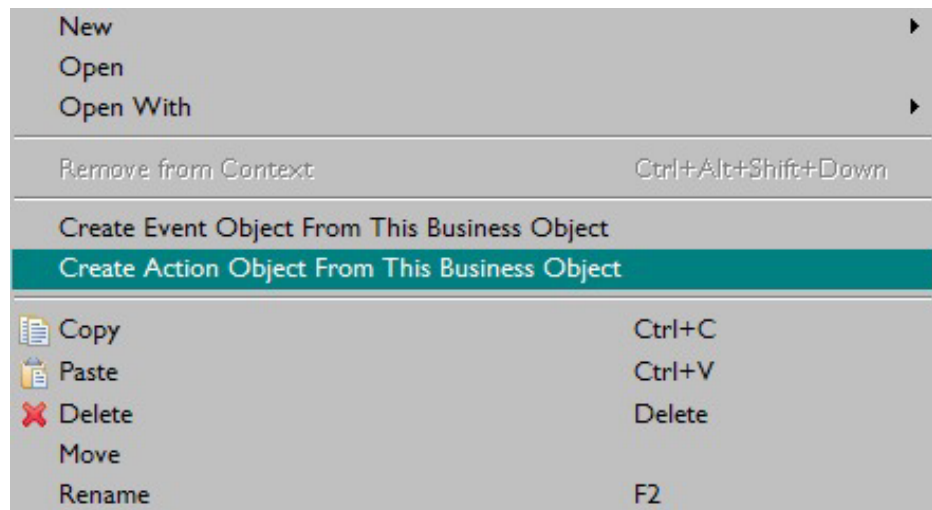
The two business objects, Customer and Car, are created in the Getting Started with Events project with the appropriate fields. Click the Request Details Event Object editor to see the Field Constructors table, which shows how each of the business object fields receives its value from its corresponding event object field.

The Field Constructors table now contains a list of the Car and Customer business object fields with the details of how the fields are defined. Each field receives its value from the corresponding event object field.

- e. Press Ctrl+S to save your updates and close the editor tabs.
- f. Create the two action objects, called Car and Customer. In the Event Explorer view, right-click the Car business object and click **Create Action Object from this Business Object**.

The New Action Object wizard opens.

The following screen capture shows the **Create Action Object from this Business Object** menu option.



- g. Name the action object Car, select Getting Started with Events as the project, and select Add To Campaign as the action. Click **Finish** to create the action object.

- h. Create another action object called Customer in the same way.

The following screen capture shows the Customer action object with the fields defined from the Customer business object.

Action object: Customer

▼ **General Information**

Name: Customer

▼ **Fields**

The fields in the action are grouped into action objects, which can be optionally reused by other actions.

Name	Type	Definition Ty...	Definition Val...	
• Last Name	String	Field	Last Name	<input type="button" value="Add..."/> <input type="button" value="Remove"/> <input type="button" value="Edit"/> <input type="button" value="Move Up"/> <input type="button" value="Move Down"/>
• First Name	String	Field	First Name	
• Zip Code	String	Field	Zip Code	
• Phone	String	Field	Phone	

- i. Close both Action Object editor tabs and save your changes.

Results

You have now defined the Website Quote Request event, the Add To Campaign action, and you have specified how the information in the event is also contained in the action.

Although creating the business objects is not necessary in this exercise, in most applications that you build, the business objects contain information from more than one incoming event. Also, the outgoing actions typically do not have a one-to-one relationship with the incoming events. In a business object, you can perform calculations by using JavaScript, obtain additional information from data connections such as a database, or obtain the result of the evaluation of additional business rules.

Next, you edit the event and action definitions to specify how they connect to the insurance company business.

Task b: Defining connections to the systems

About this task

The event that you have defined is received from a website, and the action is sent to the Marketing department. You must now configure the connection details of the website and the Marketing department system. To communicate with other systems, the event runtime uses connectors. Connectors are supplied for a range of communication protocols, including HTTP (to communicate with websites), SMTP (to send emails), File System (to send and receive data by using files in the local or remote file system), and others.

To receive the Website Quote Request events directly from the insurance company website, you have to configure an HTTP event connector. When the potential customer submits a quotation request on the website, the request, in the format of a Website Quote Request event, is sent through the HTTP event connector to the event runtime.

To send the Add To Campaign actions to the Marketing department system, you have to configure a File System action connector. When the details of the potential

customer are sent to the Marketing department in the format of an Add To Campaign action, the action is placed as a file in a directory that you specify on the file system. (For the purposes of this tutorial, the directory is on the same file system as the event runtime but you can send the action to a file system on a remote computer).

Because the connector is associated with the event or action, you must configure the connector details for each event or action that you create.

Procedure

1. Configure the HTTP event connector for the Website Quote Request event:
 - a. In the Event Explorer view, double-click the Website Quote Request event to open the editor for the event.
 - b. Click the **Connector** tab to open the Connector editor.
 - c. Select **HTTP** as the Connector Type.
 - d. Select **Express** as the Subscription type.
 - e. The information in the incoming Website Quote Request event is from a form on the website so the value **HTML form** is the correct value in the Event format field.

*Website Quote Request

Connector

Connector Type

Connector type: HTTP

Subscription type: Express Reliable

HTTP Connector Settings

Event format: HTML form

Redirect to a Web page after sending the event

URL:

e.g. http://www.example.com:9080/folder/example.html

- f. Click **Generate Sample Form**. The web page that is generated is used when you test your application.

An example of the generated form is shown in the following screen capture.

Website Quote Request

Request Details

First Name	<input type="text"/>
Last Name	<input type="text"/>
Zip Code	<input type="text"/>
Phone	<input type="text"/>
Registration	<input type="text"/>
Year	<input type="text" value="0"/>

Send Website Quote Request

- g. Press Ctrl+S to save the Website Quote Request editor updates and close the editor tab.

You have now configured the HTTP event connector.

2. Configure the File System action connector for the Add To Campaign action:

- a. On Windows, create the following folder:

C:\tutorial\Actions

On Linux, create the following folder:

/home/user/DecisionServer/Actions

Where user is your home directory. This Actions folder is where any Add To Campaign action files are put by the event runtime after processing the Website Quote Request event.

- b. In the Event Explorer view, double-click the Add To Campaign action to open the Action editor.
- c. Click the **Connector** tab to open the Connector editor.
- d. Select **File System** as the Connector type.

The information that is sent in the outgoing Add To Campaign action is in the format you defined in the Add To Campaign action, therefore **Connector packet** is the correct value in the Action format field. (The actual incoming events and outgoing actions that contain data are known as event packets and action packets).

- e. Within the **Enter the destination of the files** section, in the **Folder on that computer** field, type:

C:\tutorial\Actions

On Linux, type::

/home/user/DecisionServer/Actions

- f. In the **File pattern** field, type

Add To Campaign*.xml

This means that the File System action connector sends any file that matches that pattern of file name to the Actions directory. The asterisk (*) is replaced with a unique string of characters to ensure that all actions have unique file names and previous actions are not overwritten.

*Add To Campaign ☒

Connector

▼ Connector Type

Connector type: File System

Action packet version: 6.2

Subscription type: Express Reliable

▼ File System Connector Settings

Action format: Connector packet

Enter the destination of the files:

Computer where the IBM Decision Server connectors are installed

Other computer:

Host name:

Folder on that computer:

File pattern:
e.g. *.xml

- g. Press Ctrl+S to save the Add To Campaign editor updates and close the editor tab.

You have now configured the File System action connector.

Results

You have now completed the definition of the incoming event and outgoing action.

Task c: Building the business logic

About this task

When the potential customer requests a quotation from the insurance company website, an instance of the Website Quote Request event is received by the event runtime. The event runtime then processes the information in that event according to the business logic that has been defined. In this exercise, every time an instance of the Website Quote Request event is received, an instance of the Add To Campaign

action is sent to the Marketing department so that they can add the potential customer to relevant marketing campaigns.

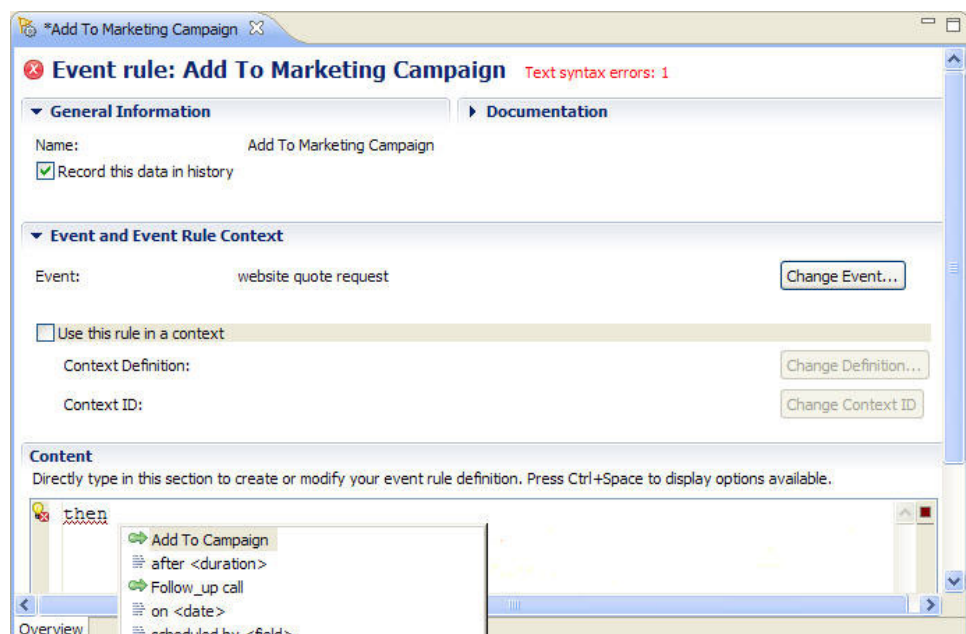
In this exercise, you define the simple business logic (the Add To Marketing Campaign event rule) that sends the Add To Campaign action in response to incoming instances of the Website Quote Request event.

An *event rule* contains the business logic that is used to process incoming instances of events. An event rule always contains a reference to an event definition. When an instance of that event is received, the event rule starts processing the event according to the business logic in the event rule. Event rules can be grouped into event rule groups so that you can maintain a set of similar rules together, or so that you can have a rule that processes the event if none of the other event rules in the event rule group starts processing that event. In this exercise, you create just one event rule. Finally, event rules usually contain a reference to at least one action definition, which is sent according to the outcome of the event processing.

Typically, an event rule contains one or more conditions against which the event is evaluated. You add conditions to this event rule in a later exercise. In this exercise, the event rule does not contain any conditions so that you can test the data model you have created, and ensure that the application works.

Procedure

1. In the Event Explorer view, right-click the Getting Started with Events project, then click **New > Event Rule**.
2. In the New Event Rule wizard, name the event rule Add To Marketing Campaign and click **Next**.
3. Select the website quote request event as the event to trigger this event rule to fire (to start processing). Click **Finish**. (The context relationship is added in a later exercise.) The Add to Marketing Campaign event rule opens in its editor.
4. In the Content section, type then and press the Spacebar. The content assist menu opens. The content assist menu lists all the valid options that you can enter in this part of the event rule. The following screen capture shows the content assist menu:



5. In the content assist menu, double-click the Add to Campaign action.
6. Double-click the ; displayed to complete the event rule.
7. Press Ctrl+S to save your changes and close the editor tab. The following screen capture shows the Add To Marketing Campaign event rule:

Event rule: Add To Marketing Campaign

▼ **General Information** ▶ **Documentation**

Name: Add To Marketing Campaign

Record this data in history

▼ **Event and Event Rule Context**

Event: website quote request

Use this rule in a context

Context Definition:

Context ID:

Content

Directly type in this section to create or modify your event rule definition. Press Ctrl+Space to

```
then Add To Campaign;
```

For now, there are no rule conditions. Every time the Website Quote Request event is received, the event rule sends the Add To Campaign action.

Results

The event project is now ready to deploy and test.

Task d: Deploying the event project

About this task

Now that you have completed the data model and business logic definitions in the application, you must deploy the event project to the event runtime which is running on your computer so that you can test that the application works.

Procedure

1. In the Event Explorer view, right-click the Getting Started with Events project and then click **Deploy**.
2. In the Deploy wizard, select **Deploy all assets** and click **Next**.
3. Check the port number of the event runtime.
 - a. Search for the AboutThisProfile.txt file.
 - On Windows: search in the *WAS_install_dir*\AppServer\profiles\ODMSample8010\logs directory in the file system.
 - On Linux: search in the *WAS_install_dir*/AppServer/profiles/ODMSample8010/logs directory in the file system.

Where *WAS_install_dir* is the installation directory for WebSphere Application Server, and where ODMSample8010 is the name of the profile.

- b. Open the AboutThisProfile.txt file and search for the line that starts with HTTP transport port. The port number for the event runtime is found on this line. By default, the value is 9080.
4. In the Deploy wizard, select **Use a temporary runtime**, then enter the details of the event runtime connection.

Host name

Enter the host name for the sample server. For example, localhost.

Port Enter the HTTP transport port number identified in step 3 on page 15. For example, 9080.

User name

Enter the WebSphere Application Server administrator user name that was specified when the sample server was installed. For example, wbeAdmin.

Password

Enter the WebSphere Application Server administrator password that was specified when the sample server was installed.

The user name and password are the authentication details that were created when WebSphere Application Server was installed.

5. Click **Finish**. The event project of assets is deployed to the run time. If you are asked for a user ID and password, enter admin for both the user ID and the password.

When the project is deployed, a Successfully deployed project to runtime message is displayed.

If the deployment fails, an error message is displayed in the **Problems** view. Check that you have defined the assets in the event project as described in the steps in this topic.

Results

You have now deployed the event project. You can complete the exercise by testing the event project.

Task e: Testing the event project Before you begin

By default, when Decision Server is installed, application server security is enabled. When using the HTTP connector, as with this example, if application server security is enabled, you must specify a user role mapping. The user role mapping determines which users have permission to send events via HTTP to the connector. To specify the user role mapping by using the WebSphere Application Server administrative console, complete the following steps:

1. Log in to the WebSphere Application Server administrative console. The default URL for the console is `http://localhost:9060/ibm/console`. The host name and port number might be different for your system.
2. In the menu, expand **Applications**, then **Application Types**. Click **WebSphere enterprise applications**.
3. In the main panel, click the HTTP connector application, **wbehttpconnector**.
4. Under Detailed Properties, click **Security role to user/group mapping**.
5. Select the **HTTPEventConnectorUser** role and map users or groups to the role. To allow unrestricted access when security is enabled, select **Map Special Subjects > Everyone**.

6. Click **Save**.

These steps are required only when an event project that uses the HTTP connector is first deployed. When subsequent projects are deployed, existing mappings are maintained.

About this task

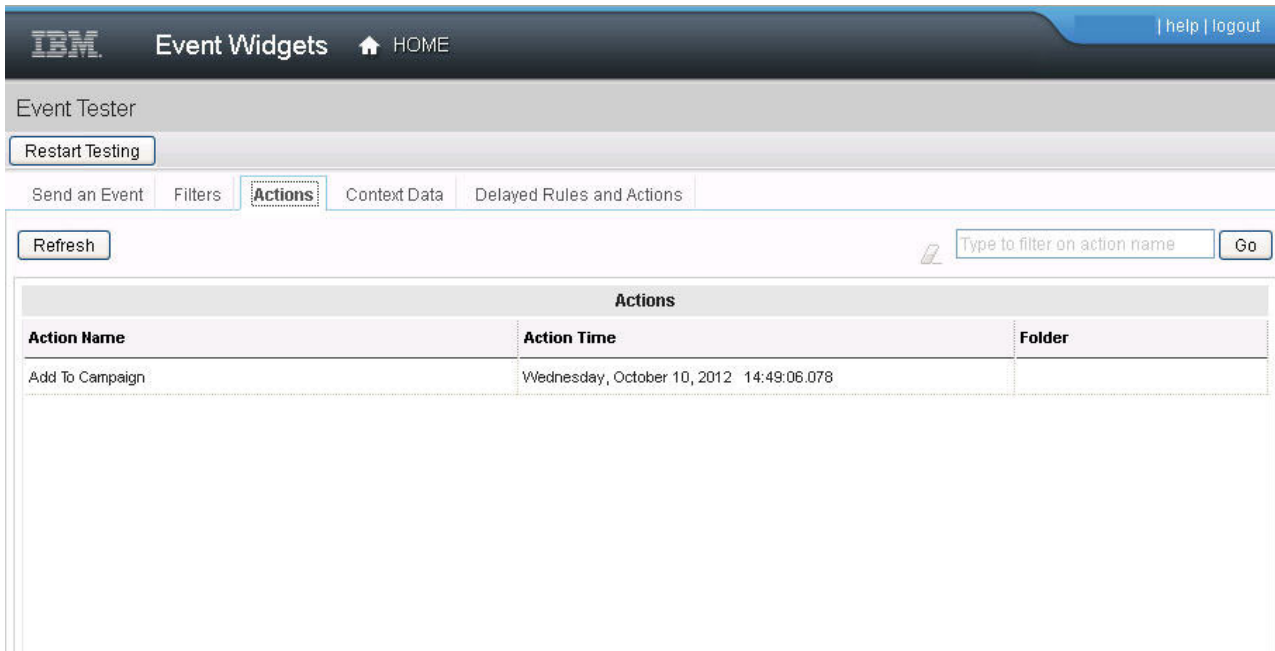
Use the web page that you generated to send Website Quote Request events to test your application. Check that the application is working correctly by using the Event Tester widget. For each Website Quote Request event that you send, your application fires an Add To Campaign action.

The HTTP connector starts automatically. If you run your application on a production system, you must start the File System connector. The File System connector is used later in the tutorial. However, by using the Event Tester widget, the other connectors are not required to test your application.

If the event runtime is not running on the default host name (localhost) and port number (9080), save the generated web page and use a text editor to edit the line in the source of the web page that specifies the host name and port number. Do not change any other details in the source of the web page

Procedure

1. In the web page that you generated earlier, enter some details in the form, then click **Send Website Quote Request**. It does not matter what the details are for this exercise. When the event is sent, the message Form processed is displayed.
2. Access the Event Tester widget:
 - a. Ensure that the steps documented in Preparing the environment for testing event logic have been completed. When Decision Server Events security is enabled, user security roles and administrative user roles must be configured correctly before you can access Event Widgets. See User roles and administrative user roles.
 - b. Access the event widgets:
 - On Windows: **Start > All Programs > IBM > Operational Decision Manager V8.5.1 > Sample Server > Events Tools > Event Widgets**
 - On Linux: **Applications > IBM > Operational Decision Manager V8.5.1 > Sample Server > Events Tools > Event Widgets**If security is enabled, the first time that you access the event widgets, you might be prompted to accept a security certificate. By default, security is enabled therefore only authorized users can access the event widgets.
 - c. Enter a user ID and password to log in to the event widgets. See Accessing Event Widgets
3. In the Event Tester widget, click the **Actions** tab to display the instances of actions that have been sent. The table lists the Add To Campaign action and the time that the action was sent in “Task a: Creating an event definition and an action definition” on page 4.



4. Look in the Actions folder:

- On Windows, look in C:\tutorial\Actions.
- On Linux, look in /home/user/DecisionServer/Actions.

Because Decision Server Events used the File System action connector to send the Add To Campaign action, there is now a file in the Actions folder containing the instance of the action. The name of the action file matches the Add To Campaign*.xml naming pattern, where the * is replaced with a unique string of characters so that all actions have unique file names and do not overwrite the previous action.

Results

You have now built and tested a basic event project. Although this event project is simplistic, it is useful to build simple logic like this (incoming events and outgoing actions with no filters or delays) after building the data model to ensure that all the event and action definitions are correctly defined, and their connections are correctly configured.

What to do next

You are now ready to proceed with Exercise 2. See “Exercise 2: Adding conditional logic to the application.”

Exercise 2: Adding conditional logic to the application

In this exercise, you add conditional logic to the application that you created in the previous exercise so that the application is more useful and sends actions only when it is appropriate to do so.

Before you begin

Complete the steps in “Getting started with event rules” on page 1.

Before starting this exercise, complete the instructions for Exercise 1. See “Exercise 1: Building a simple application” on page 3.

About this task

In the previous exercise, you built a very basic application that sent an instance of the Add To Campaign action (containing the details of the potential customer) to the Marketing department every time the insurance company received an instance of the Website Quote Request event (containing the details of the potential customer) from the insurance company website.

There are two obvious problems with an insurance company adding a potential customer to marketing campaigns every time the potential customer requests a quotation for car insurance:

- If the potential customer requests more than one quotation for their car insurance, they end up being added to the marketing campaigns more than once. This duplication is likely to annoy the potential customer and ultimately discourage them from buying a policy with this insurance company.
- If the potential customer buys the policy for which they have requested a quotation, it is best not to add them to the marketing campaigns that are intended to encourage them to buy the policy. Again, being added to irrelevant marketing campaigns might annoy the customer.

To avoid irritating an existing customer (or potential customer), you can add conditional logic to the application.

As well as creating the filters, you must identify quotation requests as either being related or as unique requests. You can define a relationship between multiple quotation requests by using various criteria but, in this exercise, you use the registration details of the car for which the insurance quotation is being requested. If more than one quotation request is received about the same car registration, the Website Quote Request events that contain that car registration are related.

In this exercise, you define how multiple instances of incoming events are related to each other, and you create the conditional logic that checks whether an instance of the Add To Campaign action has already been sent for a specific car. Follow tasks a-c to complete exercise 2.

Task a: Writing a condition in the event rule

About this task

When the conditions in an event rule are evaluated, they return a response that is either true or false.

The condition you are about to write in the Add To Marketing Campaign event rule checks that the customer has not already been added to the marketing campaign.

This is done by checking whether there have been any previous occurrences of the Add To Campaign action. If there have not been any, it is safe to add the potential customer to the marketing campaign by sending the Add To Campaign action.

Procedure

1. In the Event Explorer view, double-click the Add To Marketing Campaign event rule to open the Event Rule editor. Place the cursor immediately in front of the

word then and press **Enter**. On the empty line inserted above, type: `if`. Press the **Ctrl+Spacebar** to display the content assist menu.

2. On the content assist menu, double-click (or move down the menu and press **Enter**) all occurrences of.

The text `all occurrences of` is added to the event rule. Also, another menu is displayed to prompt you to select the event or action to include in this condition.

3. Double-click `add to campaign` to add this action to the event rule.

The next menu prompts you to complete the event rule condition.

4. Double-click `is <an object>`. When prompted, double-click `<number>`. A `0` (zero) is added to the event rule. The event rule is now complete.

The following screen capture shows the Add To Marketing Campaign event rule condition:

The screenshot shows the configuration page for an event rule named "Add To Marketing Campaign". At the top, there is a red error message: "Event rule: Add To Marketing Campaign An Event Rule, which is not in a context, has a condition that requires a context." Below this, the interface is divided into several sections:

- General Information:** Name: Add To Marketing Campaign; Record this data in history.
- Event and Event Rule Context:** Event: website quote request (with a "Change Event..." button); Use this rule in a context; Context Definition: (with a "Change Definition..." button); Context ID: (with a "Change Context ID..." button).
- Content:** A text area containing the rule definition: `if all occurrences of add to campaign is 0 then Add To Campaign ;`

When the context relationship is defined later in the exercise, the context error message that you see is resolved.

5. Save the event rule.

Task b: Verbalizing a business object field to define a context relationship

About this task

When an instance of an event is received and the definition of that event is referred to by one or more event rules, the occurrence of that event instance is recorded. This record means that the event rule can check whether an instance of the event has previously been received.

When a potential customer requests a quotation then requests a second quotation, the event rule must somehow identify that these two requests (two instances of the Website Quote Request event) are related. Otherwise, the event rule might identify all requests as unique and the potential customer who has requested two quotations is identified as two separate potential customers. This leads to the potential customer being added to marketing campaigns twice, and with no way to identify when a customer who has previously requested a quotation has subsequently purchased the policy.

For example, a potential customer might request a quotation to insure a car of registration ABX523, then request another quotation for a car of registration ABX523. Because car registrations are unique, the two quotations are definitely for

the same car. Therefore, if the context relationship in the event rule is defined as the value in the Registration field of the incoming event, then the event rule can identify that the two quotation requests are related and the later quotation request is the second request for the same car.

This feature is important to understand when creating conditions. For example, in the condition that you just created, the event rule cannot count the number of times that quotations have been received for the same car, without knowing it has to relate the quotations by car registration.

When the Add To Marketing Campaign event rule is open in the editor, if you click **Change Context ID**, the Define the Context ID dialog box opens and displays the available business object fields.

You want to add the Registration field of the Car business object as the context relationship in this event rule. However, because we are going to change the Registration field verbalization, click **Cancel**.

Verbalization is a way to provide a more meaningful name for assets and fields so that the event rules can be understood more easily. The verbalization of events, actions, filters, business objects, business object fields, and named constants is created automatically but you can edit the verbalizations if you want to. You saw the verbalizations of the name of the Website Quote Request event and the Add To Campaign action on the content assist menu when you wrote the Add To Marketing Campaign event rule.

The verbalization for the business object fields are created automatically when you create the business object fields that you want to use in an event rule. When you created the Add To Marketing Campaign event rule, you were able to select the names of the Website Quote Request event and the Add To Campaign action from the content assist menu. By default, name verbalizations are created automatically for events, actions, filters, business objects, business object fields, and named constants.

Although the Registration field of the Car business object and the Car business object have verbalizations, edit the verbalization as described. Then set the Registration field as the context relationship for the Add To Marketing Campaign event rule.

Procedure

1. Change the Registration field verbalization of the Car business object:
 - a. In the Event Explorer view, double-click the Car business object to open the Business Object editor.
 - b. In the Fields table, double-click Registration (or click Registration and click **Edit**). The Registration field details are displayed on the Field page of the editor.
 - c. Review the **Verbalization** section of the editor window.
By default, the verbalization created is the registration of a car. In English, this phrasing is awkward. The phrasing would be clearer if it read "the car registration".

To improve the phrasing, you can change the template used to build the phrase.

- d. In the **Template** field, change the text from {registration} of {this} to {this} {registration}. When you press Ctrl+S, the verbalization phrase is changed to a car registration.

The Registration field verbalization is updated. The following screen capture shows the updated Registration field verbalization.

▼ **Verbalization**

[Edit](#) the subject used in phrases

Navigation: a car registration

Template:

[Reset](#) the verbalization to default.

[Remove](#) the verbalization.

- e. Close the editor tab and save your changes.
2. Add a context relationship to the Add To Marketing Campaign event rule.
 - a. Switch to the Add To Marketing Campaign Event Rule editor.
 - b. In the editor, click **Use this rule in a context**.
 - c. Click **Change Definition**, then select **System Context**.
 - d. Click **Change Context ID**.
 - e. Select **the car registration**, then click **Finish**.

The Add To Marketing Campaign event rule is now complete.

The following screen capture shows the Add To Marketing Campaign event rule with the context defined:

Event rule: Add To Marketing Campaign

▼ **General Information** ► **Documentation**

Name: Add To Marketing Campaign

Record this data in history

▼ **Event and Event Rule Context**

Event: website quote request Change Event...

Use this rule in a context

Context Definition: System Context Change Definition...

Context ID: the car registration Change Context ID

Content

Directly type in this section to create or modify your event rule definition. Press Ctrl+Space to display options available.

```
if all occurrences of add to campaign is 0
then Add To Campaign ;
```

3. Save the updates.

Task c: Testing the context relationship between events

About this task

You can see how the context relationship works by sending multiple instances of the Website Quote Request event and changing the value of the Registration so that sometimes it is the same as previous instances of the event, and sometimes it is unique. Observe if an instance of the Add To Campaign action is sent.

Procedure

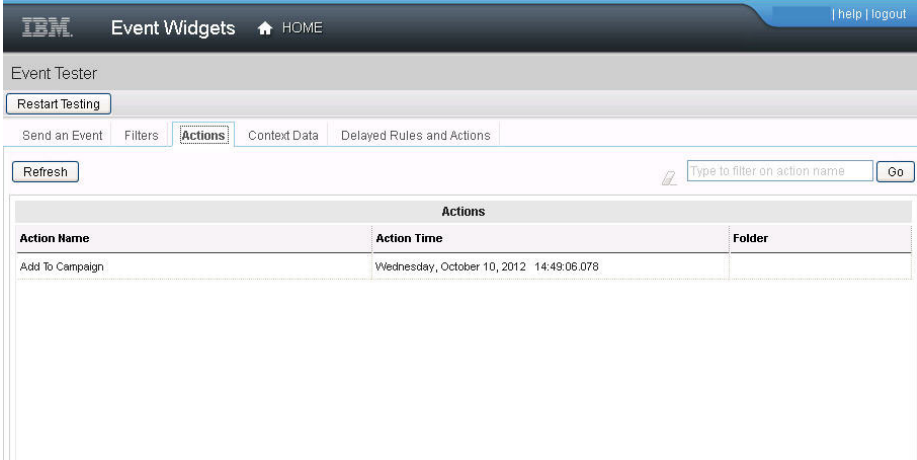
To test the context relationship:

1. Deploy the event project. In the Event Explorer view, right-click the Getting Started with Events project and then click **Deploy**. Select **Deploy all assets** and click **Finish**.

The assets contained in the updated project overwrites the previous assets contained in the project that you deployed in Exercise 1.

You can continue to send events from the web form, or you can send events by using the Event Tester widget. (In the Event Tester widget, click **Send Event**, then select the Website Quote Request event template to use).

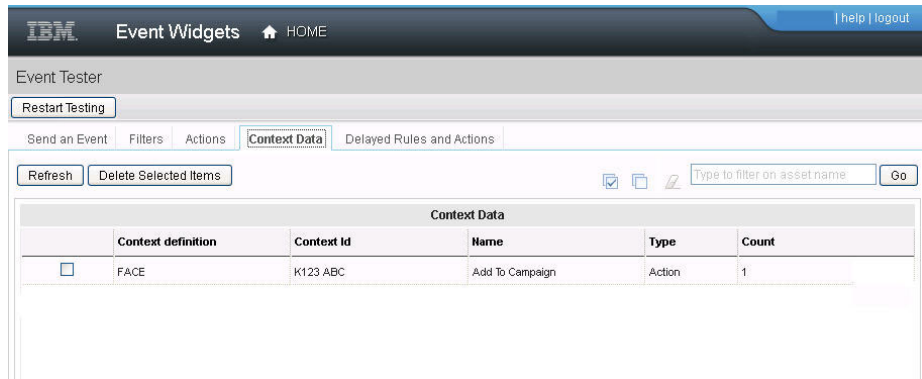
2. In the Event Tester widget, click **Restart Testing** to clear the previous event that you sent in Exercise 1 and the action that was fired as a result.
3. If you have not previously enabled the event runtime to record history, do that now. See Configuring the event runtime to record history.
4. Send the Website Quote Request once.
5. Use the Event Tester widget to see what happened when the Website Quote Request event was received:
 - a. Click the **Actions** tab. The table shows that the Add To Campaign action was sent in response to the Website Quote Request event.



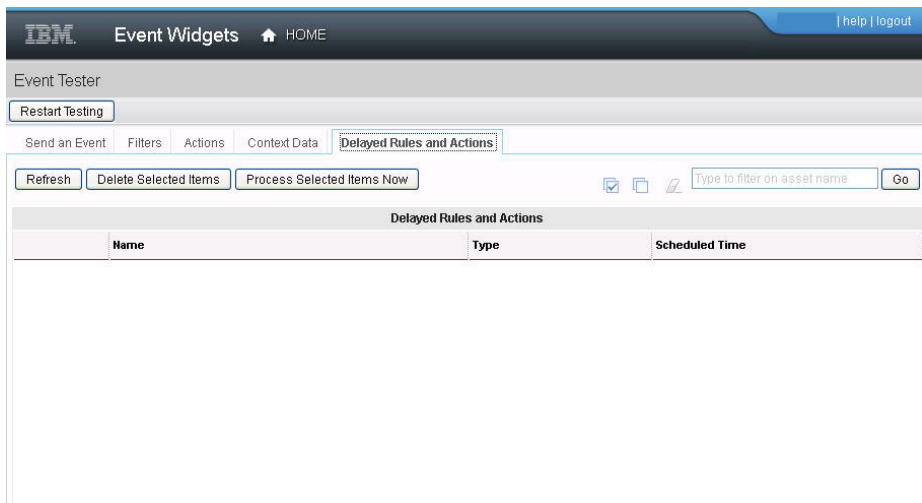
The screenshot shows the IBM Event Widgets Event Tester interface. The top navigation bar includes the IBM logo, 'Event Widgets', a home icon, and 'HOME'. There is a 'help | logout' link in the top right. Below the navigation bar, the 'Event Tester' section has a 'Restart Testing' button. A toolbar contains 'Send an Event', 'Filters', 'Actions' (selected), 'Context Data', and 'Delayed Rules and Actions'. Below the toolbar, there is a 'Refresh' button and a search box with the placeholder text 'Type to filter on action name' and a 'Go' button. The main content area displays a table titled 'Actions' with the following data:

Action Name	Action Time	Folder
Add To Campaign	Wednesday, October 10, 2012 14:49:06.078	

- b. Click the **Context Data** tab to display a list of the actions with their context information. The table shows the Add To Campaign action was sent with a context of K123 ABC (the value in the Registration field of the form).



- c. Click the **Delayed Events and Actions** tab. The table shows that there are no delayed events and no delayed actions, because you did not set any delays in the Add To Campaign event rule.



6. Submit the Website Quote Request again (without changing any of the values).
7. Check the outcome of sending the event. Look in the **Action** tab of the Event Tester widget again. This time, there are no new instances of the Add To Campaign action listed. This occurs because the event rule group has recognized that there was previously an Add To Campaign action sent in response to a request for the same car registration.
 - The Actions tab does not show any additional actions because the Add To Marketing Campaign event rule ensured that if the Website Quote Request is received a second time for the same car registration the Add To Campaign action is not sent again.
 - The Context Data tab shows that a second Website Quote Request event was received but no additional Add To Campaign action was sent.
 - Again, the Delayed Events and Action tab displays no information because there is no delay set on the event rule group.
8. Change the value in the Registration field to K123 DEF in the Event Tester widget and then send again.
9. Check the outcome of sending the event:
 - The **Actions** tab shows that the Add To Campaign action has been sent again.
 - The **Context Data** tab shows that one event has been received and one action set in the new context (K123 DEF).

- Again, there are no delayed events or actions shown on the **Delayed Events and Actions** tab.

Results

In the next exercise, you create a filter that checks that the customer has not already bought a policy.

What to do next

You are now ready to proceed with Exercise 3. See “Exercise 3: Defining another event and building another condition.”

Exercise 3: Defining another event and building another condition

In this exercise, you define another event called `Policy Purchased`. You also add another condition to the `Add To Marketing Campaign` event rule so that the event rule can check whether a policy has already been purchased for the car. If a policy has been purchased, the customer is not added to marketing campaigns unnecessarily. As an application evolves, you might typically find that the event project has to be updated by the IT user occasionally before the business user can continue building the business logic.

Before you begin

Complete the steps in “Getting started with event rules” on page 1.

Before starting this exercise, complete the instructions for Exercise 2. See “Exercise 2: Adding conditional logic to the application” on page 18.

About this task

In this exercise, you add a filter to the `Add To Marketing Campaign` event rule. The new filter, called `Has not purchased a policy`, checks for an instance of the `Policy Purchased` event being received in the past year for the same car registration (as defined by the context relationship in the event rule).

Follow tasks a-e to complete exercise 3.

Task a: Define the policy purchased event

After a customer has requested a quotation for car insurance, if the customer purchases the insurance policy, an instance of the `Policy Purchased` event is received by the policy system of the insurance company.

Procedure

1. In the `Getting Started with Events` project, create a blank event called `Policy Purchased`.

The event object for this event is the same as the event objects for the `Website Quote Request` event. You can define the `Policy Purchased` event to share the same event object as the `Website Quote Request` event.

2. In the `Policy Purchased Event Object` editor, click **Add** to open the `New event object wizard`.
3. In the wizard, click **Share existing event object** and click **Next**.

4. Expand **Event Objects** and select the Request Details event object from the list. Click **Finish**.
5. Click the **Connector** tab to open the Connector editor.
6. Select **File System** as the Connector type.
7. Check that the Subscription type is **Express**.
8. Check that **Connector packet** is selected for the Event format.
9. Within the **Enter the destination of the files** section, in the **Folder on that computer** field, type:
 On Windows, C:\tutorial\Events
 On Linux, /home/user/DecisionServer/Events
10. In the File pattern field, type
 Policy Purchased*.xml

This means that the File System event connector sends any file that matches that pattern of file name to the Events directory. The asterisk (*) is replaced with a unique string of characters to ensure that all events have unique file names and previous events are not overwritten.

11. Save the Policy Purchased event.

The Request Details event object is now also displayed under the Policy Purchased event in the Event Explorer view. If you make changes to the Request Details event object, the changes will affect both events.

The following screen capture shows the fields defined in the Request Details event object.

Event object: Request Details

General Information

Documentation

Name: Request Details

Fields

Number of Occurrences

Add fields to this event object

Name	Type	Used
• First Name	String	True
• Last Name	String	True
• Zip Code	String	True
• Phone	String	True
• Registration	String	True
• Year	Integer	True

Specify the number of instances of this event should appear in the event.

Unlimited

0 Minimum number of occurrences

0 Maximum number of occurrences

Field Constructors

Add business objects that will receive data from the fields in this event object:

Business Object	Field	Data Type	Definition Type	Definition Value
• Car	Year	Integer	Field	Year
• Car	Registration	String	Field	Registration
• Customer	Phone	String	Field	Phone
• Customer	Zip Code	String	Field	Zip Code
• Customer	Last Name	String	Field	Last Name
• Customer	First Name	String	Field	First Name

12. Close the editor tab and save your changes. You have now defined the Policy Purchased event. The Policy Purchased event definition contains the same fields of data as the Website Quote Request event definition.

Next, you create some conditional logic to check whether the Policy Purchased event has already been received.

Task b: Create the filter

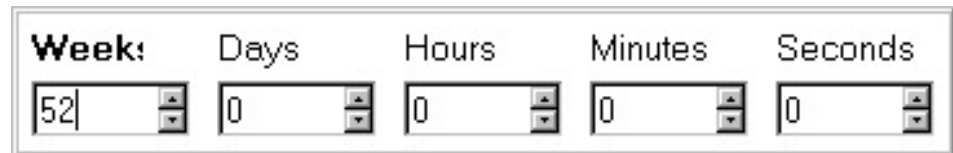
Create the Has not purchased a policy condition as a separate, reusable filter that you can reference from both of the event rules.

About this task

The conditional logic that you need to add to the event rule is also required in another event rule in Exercise 4. Therefore, it is useful to create this condition as a separate, reusable filter that you can reference from both of the event rules. The Has not purchased a policy filter checks whether an insurance policy for the car has already been purchased. If a policy has already been purchased, the insurance company does not want to add the customer to the marketing campaigns. Because car insurance policies are typically valid for 12 months, the filter checks whether a policy has been purchased for the car in the past 52 weeks.

Procedure

1. In the Event Explorer view, right-click Getting Started with Events, then click **New > Filter**. Name the filter Has not purchased a policy. Click **Finish**.
You now define the logic of the filter to check that the potential customer who submitted the quotation request has not purchased a policy in the last year.
2. In the Content section of the Filter editor, press Ctrl+Spacebar to display the content assist menu.
3. From the content assist menu, double-click past occurrences of <an event or action>. A list of available events is displayed.
4. Double-click policy purchased.
5. Double-click within <duration> and then double-click <duration>. A dialog box containing different time options is displayed.
6. In the **Weeks** field, type 52 and press **Enter**.



The image shows a dialog box for selecting a time duration. It has five input fields: Week, Days, Hours, Minutes, and Seconds. The Week field contains the number 52, while the other fields contain 0. Each field has a small spinner icon to its right.

7. After 52 weeks, press Ctrl+Spacebar . From the content assist menu, double-click is <an object>. You can enter the number of past occurrences of the Policy Purchased event that the event rule checks for.
8. Complete the expression by entering a number. Type 0 (zero). Press Ctrl+S to save your changes. The complete expression now reads:
past occurrences of policy purchased within 52 weeks is 0

This means that the filter evaluates to true if the number of occurrences of the Policy Purchased event in the past 52 weeks is exactly zero.

Check the **Problems** view.



The image shows a screenshot of the Problems view in an IDE. The title bar indicates 'Tasks', 'Problems', 'Event Table', and 'Event Runtimes'. The status bar shows '0 errors, 1 warning, 0 others'. A table lists the warning:

Description	Resource	Path	Location	Type
Warnings (1 item)				
Makes a complex reference to EventPolicyPurchased which is not triggered by any existing event rule and si	Has not purcha...	/Getting Started with ...	Content	Events Problem

A warning message is displayed for the event rule. This warning occurs because the event that it refers to, Policy Purchased, is not used as the

triggering event for an existing event rule. This means that the Policy Purchased event is never recorded so it cannot be associated with any other instances of the event.

9. In the Verbalization section, change the generated Phrase to lowercase has not purchased a policy.
10. Save the filter.

Task c: Add the filter to the event rule

The Has not purchased a policy filter checks whether an insurance policy for the car has already been purchased.

Procedure

1. In the Add To Marketing Campaign Event Rule editor, press **Enter** at the end of the first line to insert a blank line.
2. On the new blank line, type and and press Ctrl+Spacebar.
3. In the content assist menu, double-click has not purchased a policy.
4. Save the event rule changes. A reference to the Has not purchased a policy filter is added to the Add To Marketing Campaign event rule.

Task d: Create the event rule

Remove the warning in the **Problems** view.

Before you begin

In the **Problems** view, the event rule that you have just created is still marked with a warning: Makes a complex reference to Event Policy Purchased which is not triggered by any existing event rule and so will never occur. (BEER1255W).

This warning occurs because the old condition references the Policy Purchased event definition, which is not used directly in any event rule. Because there are no event rules to respond to an incoming instance of the Policy Purchased event, instances of the Policy Purchased event are not recorded. Therefore, any conditions that reference the Policy Purchased event are not evaluated correctly, which is the reason why the Add To Marketing Campaign event rule is still marked with a warning.

You now create an event rule, Add To Campaign Next Year , which responds to an incoming instance of the Policy Purchased event. In this exercise, the Add To Campaign Next Year event rule does not send out any actions. In Exercise 4, you modify this event rule to send out the Add To Campaign action after a 48 week delay (that is, one month before the purchased policy expires).

Procedure

1. Create a new event rule called Add To Campaign Next Year. Click **Next**.
2. In the New Event Rule wizard, click the Policy Purchased event and click **Next**.
3. Select **System context**, then click **Next**.
4. To define the context ID, select the car registration and click **Finish**.
The Add To Marketing Campaign event rule is now valid and there are no reported errors or warnings in the **Problems** view.

Task e: Deploy and test the event project and the event rule

Deploy and test the updated event project and the new Add To Campaign Next Year event rule.

Procedure

1. In the Event Explorer view, right-click the Getting Started with Events event project and click **Deploy**. Select **Deploy all assets** and click **Finish**.
2. Check the **Problems** view to ensure that there were no problems during deployment. The changes that you made to the event rule, the business objects, the new filter that you created, and the new event that you defined are all deployed to the server. When you deploy the contents of the event project, the previous project assets are overwritten.
3. In the Event Tester widget, click **Restart Testing** to clear the events that you sent in Exercise 2.
4. To test the changes you have made, send the following sequence of test events, checking each time for an action according to what is listed in the following table:

Table 2. Events to send and details to check

Step	Event	How to submit the event instance	Action	Reason for action
1	Website Quote Request	Use the Event Tester widget with a new value in the Registration field, for example K123 GHI.	Add To Campaign	This instance is the first time the Website Quote Request event is received with the new value you entered in the Registration field. That is, the first request for a quotation for this car. Therefore, the details of the potential customer are forwarded (in the Add To Campaign action instance) to the Marketing department.
2	Website Quote Request	Use the Event Tester widget with the same value in the Registration field as in Step 1.	No action.	This instance is the second time the Website Quote Request event is received for the car registration used in Step 1. That is, the second request for a quotation for a car. Therefore, the details of the potential customer are <i>not</i> sent to the Marketing department again.

Table 2. Events to send and details to check (continued)

Step	Event	How to submit the event instance	Action	Reason for action
3	Policy Purchased	Use the Event Tester widget to send an instance of the Policy Purchased event with a new value in the Registration field, such as ZXY 456. Click the Send an Event tab, then select Policy Purchased. Finally, click Send Event .	No action.	The arrival of an instance of the Policy Purchased event only shows that a potential customer has purchased a policy. Therefore, no action is required.
4	Website Quote Request	Use the Event Tester widget with the same value in the Registration field as in Step 3.	No action.	This time, when an instance of the Website Quote Request event is received for the same car registration as in Step 3, the existence of the Policy Purchased event from Step 3 indicates that the customer has already bought a policy for this car. Therefore, there is no reason to add them to the marketing campaign yet.

What to do next

You are now ready to proceed with Exercise 4. See “Exercise 4: Creating event rules with delays.”

Exercise 4: Creating event rules with delays

In this exercise, you create another event rule that contains a delay so that the incoming event is not processed by the event rule until a specific time period after the event has arrived. You also update the Add to Campaign Next Year event rule with a delay of 48 weeks.

Before you begin

Complete the steps in “Getting started with event rules” on page 1.

Before starting this exercise, follow the instructions for Exercise 3. See “Exercise 3: Defining another event and building another condition” on page 25.

About this task

When a potential customer requests a car insurance quotation, you can add the potential customer to a marketing campaign that can then be customized to encourage the customer to consider buying the policy. Often, though, potential customers request two (or even more) quotations for car insurance; for example, if they are hunting around different insurance companies for a suitable insurance policy and return to your company for a second quotation after having requested a quotation that they liked six weeks earlier.

If a potential customer has requested two or more quotations for car insurance, there is a good chance that if the insurance company phones the potential customer to check whether there is anything they can do to help the potential customer purchase one of their policies.

Of course, it is more expensive for the car insurance company to have someone phone up a potential customer in person. Therefore, it is important to identify the particular situations where a personal phone call is likely to make the sale that might otherwise not be made.

You can create an event rule that helps identify such situations.

In this exercise, you create an event rule called `Make Follow_up Call`. When an instance of the `Website Quote Request` event is received, the `Make Follow_up Call` event rule checks whether the potential customer has already purchased a policy in the past year. Also, whether the potential customer has already requested a quotation in the past 12 weeks. If both of these conditions are true, an instance of the `Follow Up Call` action is sent to the company call center with the customer details so that a call center representative can phone the potential customer.

You also update the `Add To Campaign Next Year` event rule so that customers who buy the policy are added to a marketing campaign in 48 weeks time, which is one month before they must renew their policy. To implement this condition, you add a delay of 48 weeks to the `Add To Campaign` action in the event rule.

Follow tasks a-e to complete Exercise 4.

Task a: Define the action

Procedure

1. Create an action called `Follow_up Call`.
2. In the Action editor, add the existing action object called `Customer` to the `Follow_up Call` action. Click **Add** and select the **Share existing action object** option.
3. Save the editor changes.

The `Customer` action object, which you created for the `Add To Campaign` action, is now used by the `Follow_up Call` action too.




The `Customer` business object is already mapped to the `Customer` action object so the `Follow_up Call` action is now complete.
4. In the Action editor of the `Follow_up Call` action, click the **Connector** tab. Select **File System** as the Connector type.
5. In the File System Connector Settings section, check that the Action format field says **Connector packet**.
6. In the **Folder on that computer** field, type:
 - On Windows, `C:\tutorial\Actions`

- On Linux, /home/user/DecisionServer/Actions
7. In the File pattern field, type Follow_up Call*.xml
 8. Save the editor changes. Your changes are saved to the local file system.

Results

You have now defined the Follow_up Call action.

Action: Follow_up Call

<p>General Information</p> <p>Name: Follow_up Call</p>	<p>Documentation</p> <p>Verbalization</p> <p>Navigation:</p> <p>Phrase: <input type="text" value="follow_up call"/></p> <p>Action:</p> <p>Template: <input type="text" value="follow_up call"/></p> <p> Reset the verbalization to default.</p> <p> Remove the verbalization.</p>
<p>Action Objects</p> <p>The fields in the action are grouped into action objects, which can be optionally reused by other actions.</p> <div style="border: 1px solid black; padding: 5px;"> <p> Customer</p> </div> <div style="float: right; margin-top: 10px;"> <p><input type="button" value="Add..."/></p> <p><input type="button" value="Remove"/></p> <p><input type="button" value="Edit"/></p> </div>	

Task b: Create the event rule

Procedure

1. Create a new event rule called Make Follow_up Call.
2. In the New Event Rule wizard, select the website quote request event to be the event that the event rule processes. Click **Next**.
3. Select the the car registration field as the context relationship for this event rule. Click **Finish**.
4. In the Content section, type if and press Ctrl+Spacebar.
5. In the content assist menu, double-click has not purchased a policy.
6. Double-click then <actions> and double-click the follow_up call action.
7. Double-click the ; displayed in the content assist menu to complete the event rule.
8. Press Ctrl+S to save the event rule.

Results

You have now made the Make Follow_up Call event rule.

Event rule: Make Follow_up Call

General Information | **Documentation**

Name: Make Follow_up Call
 Record this data in history

Event and Event Rule Context

Event: website quote request

Use this rule in a context

Context Definition: System Context

Context ID: the car registration

Content
Directly type in this section to create or modify your event rule definition. Press Ctrl+Space to display options available.

```
if has not purchased a policy then follow_up call ;
```

Task c: Create another condition in the event rule Procedure

1. In the Make Follow_up Call Event Rule editor, insert a new line before the has not purchased a policy filter.
2. After if, press Ctrl+Spacebar.
3. On the content assist menu, double-click past occurrences of <an event or action>.
4. Double-click this event.
5. Double-click within <duration>.
6. Double-click <duration>.
7. In the weeks box of the duration dialog, type 12, then press **Enter**.
8. After 12 weeks press Ctrl+Spacebar and then double-click is <an object>.
9. Double-click <number>, and then type 1.
10. Double-click and <condition>.
11. Press Ctrl+S to save the editor changes.

Results

The following screen capture shows the Make Follow_up Call event rule with the added delay.

Event rule: Make Follow_up Call

▼ General Information		▶ Documentation	
Name: Make Follow_up Call			
▼ Event Rule Context and Event			
Context ID:	<input type="text" value="the car registration"/>	<input type="button" value="Change Context..."/>	
Event:	<input type="text" value="website quote request"/>	<input type="button" value="Change Event..."/>	
Content			
Directly type in this section to create or modify your event rule definition. Press Ctrl+Space to display options available.			
<pre>if past occurrences of this event within 12 weeks is 1 and has not purchased a policy then follow_up call ;</pre>			

Task d: Update the event rule About this task

Update the Add To Campaign Next Year event rule by adding a delayed action to add the customer to the marketing campaign one month before the purchased policy expires

Procedure

1. Open the Add To Campaign Next Year event rule in the editor.
2. In the Content section, type after and then press Ctrl+Spacebar.
3. On the content assist menu, double-click <duration>.
4. In the weeks field of the dialog displayed, type 48 and then press **Enter**.
5. Position the cursor after the 48 weeks and press Ctrl+Spacebar. Double-click then <actions>.
6. Double-click add to campaign.
7. Double-click ;.
8. Save your updates.
9. Check the **Problems** view for errors.

Results

You have now updated the Add to Campaign Next Year event rule.

Event rule: Add To Campaign Next Year

▼ General Information		▶ Documentation	
Name: Add To Campaign Next Year			
▼ Event Rule Context and Event			
Context ID:	<input type="text" value="the car registration"/>	<input type="button" value="Change Context..."/>	
Event:	<input type="text" value="policy purchased"/>	<input type="button" value="Change Event..."/>	
Content			
Directly type in this section to create or modify your event rule definition. Press Ctrl+Space to display options available.			
<pre>after 48 weeks then add to campaign ;</pre>			

Task e: Deploy and test the event project and the event rule Procedure

Deploy the event project by right-clicking the project name and then click **Deploy**. Select **Deploy all assets** and click **Finish**.

To test the changes you have made, send the following sequence of test events checking each time for an action according to what is listed in the following table. For each event that you send, look at the **Filters** and **Context Data** tabs of the Event Tester widget to see which filters were evaluated and to check which events have been received and which action instances sent by the event runtime. Look at the **Delayed Rules and Actions** tab to see the delayed event. The delayed action remains on this tab for 48 weeks. After this period of delay has ended, the event is no longer listed on the **Delayed Rules and Actions** tab and is instead listed on the **Context Data** tab.

Table 3. Events to send and details to check

Step	Event	How to submit the event instance	Action	Reason for action
1	Website Quote Request	Use the Event Tester widget to send a new value in the Registration field, for example K123 JKL.	Add To Campaign	This instance is the first time the Website Quote Request event is received with the new value you entered in the Registration field. That is, the first request for a quotation for this car. Therefore, the details of the potential customer are forwarded (in the Add To Campaign action instance) to the Marketing department.
2	Website Quote Request	Use the Event Tester widget to send the same value in the Registration field as in Step 1.	Follow_up Call	This instance is the second time the Website Quote Request event is received for the car registration used in Step 1. That is, the second request for a quotation for a car. Therefore, the details of the potential customer are sent to the call center so that a follow-up call can be made to close the sale.

Table 3. Events to send and details to check (continued)

Step	Event	How to submit the event instance	Action	Reason for action
3	Website Quote Request	Use the Event Tester widget to send the same value in the Registration field as in Steps 1 and 2.	No action	This instance is the third time the Website Quote Request event is received for the car registration used in Step 1. That is, the third request for a quotation for a car. Therefore, the details of the potential customer are <i>not</i> sent to either the Marketing department or the call center again.
4	Policy Purchased	Use the Event Tester widget to send an instance of the Policy Purchased event with the same value in the Registration field as in Steps 1 - 3.	No action on the Actions tab but the Add To Campaign action is listed on the Delayed Rules and Actions tab with a delay of 48 weeks. Add To Campaign is not listed in the Actions tab until 48 weeks have gone by.	The arrival of an instance of the Policy Purchased event just shows that a potential customer has purchased a policy. Therefore, there is no action required.
5	Website Quote Request	Use the Event Tester widget to send the same value in the Registration field as in Steps 1 - 4.	No action	This time, when an instance of the Website Quote Request event is received for the same car registration as in Step 3, the existence of the Policy Purchased event from Step 3 indicates that the customer has already bought a policy for this car. This event means that there is no reason for adding them to the marketing campaign yet.

What to do next

You have now completed the Getting started with Events tutorial exercises. To learn more about Decision Server Events, see:

- Tutorial: Calculating averages to identify event patterns
- Tutorial: Comparing an event with a previous event
- Tutorial: Tracking the state of something
- Tutorial: Maintaining a running total

Tutorial: Calculating averages to identify event patterns

This tutorial shows you how to calculate the average of values in events over a specified time period. You can use this average to identify event patterns, such as customer spending habits or unusual withdrawals from a bank account.

In this tutorial, you create a small project to calculate the average cost of purchases that a customer makes in a department store so that the store can provide loyalty awards to the customer.

Before you start

In this tutorial, you create a small project to calculate the average cost of purchases that a customer makes in a department store so that the store can provide loyalty awards to the customer.

In this tutorial you work on the following tasks:

- “Task 1: Creating an event project” on page 40
- “Task 2: Defining the business objects” on page 42
- “Task 3: Defining the event rule and filters” on page 47
- “Task 4: Deploying the application” on page 50

What you will learn

In this tutorial, you will learn these skills:

- Use Decision Server Events to calculate the average of event values in a specified time period.
- Use the average value to identify event patterns such as customer spending habits.
- Track the purchases made by the customer using Decision Server Events.
- Perform a specific activity based on the results of running the event project.

This tutorial takes about 2 hours to complete.

What you need to know

This tutorial is intended for developers who want to learn how to develop event projects in Decision Server Events. The procedures in the tutorial assume that you are familiar with the Eclipse environment, and that you have completed the Getting started with event rules tutorial.

 **Start this tutorial:** “Task 1: Creating an event project” on page 40.

Tutorial scenario

The average of event values in a specified time period can be calculated in Decision Server Events. In this tutorial, you create a small project to calculate the average cost of purchases that a customer makes in a department store. The store uses the average to provide loyalty awards to a specific set of customers.

A department store generates an event each time a customer purchases one or more items in the store. The event contains the total value of the purchase. For each customer, Decision Server stores the value of each of the last five purchases, and then calculates the average of the purchases. If the customer spent an average of \$50 or more per purchase for the last five purchases, then the customer qualifies for a loyalty award. The store periodically sends award vouchers to the customers who qualify for the loyalty award.

To complete the tutorial, you complete these tasks:

1. Create an event project and add an order processing event and an action to send a loyalty award.
2. Define business objects to hold the values supplied in the event.
3. Define business object fields to contain data such as the transaction value and date, customer details, and loyalty award amount.
4. Create filters to calculate the average value of the transactions, track the number of transactions, and check to see if the relevant action has already been triggered.
5. Define the event rule that tracks the transactions and triggers the action if all of the filter conditions are met.
6. Deploy the application created by the tutorial.

For each customer, Decision Server Events builds an array in a business object using values from the last five **Order processing complete** events. The events generated by a specific customer are associated with each other because the customer ID field from each event is used as the context ID in the array. The array is stored in the **Order totals** business object. The **Customer details** business object contains the customer ID field, and also a field that contains the loyalty award details.

 **Start this tutorial:** “Task 1: Creating an event project” on page 40.

Related information:

“Running the completed tutorial”

You can import and run the tutorial in its completed state. When completed, the tutorial contains an event that you can run to calculate the average of a series of purchase values.

Running the completed tutorial

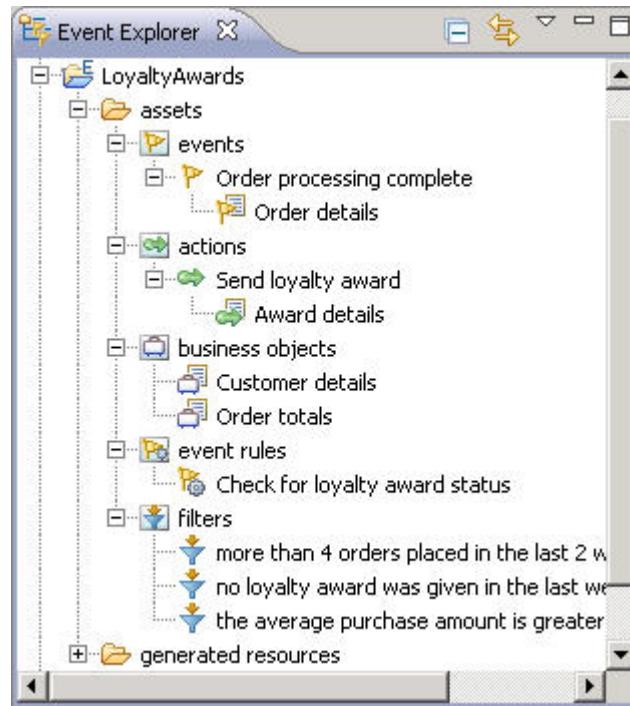
You can import and run the tutorial in its completed state. When completed, the tutorial contains an event that you can run to calculate the average of a series of purchase values.

To import the completed calculating averages project and run the tutorial application:

1. Download the completed tutorial project for calculating averages. Right-click `LoyaltyAwards.xml` and save the file to your local system.
2. In the Event Explorer view, right-click and select **Import**.

3. In the Import window, click **Event Designer > Event Project from XML File**, then click **Next**.
4. Click **Browse** to select the event project file to import: LoyaltyAward.xml.
5. Navigate to the location of the LoyaltyAwards.xml file, select the file, click **Open**, then click **Next**.
6. Click **Next** in the Project Assets panel.
7. If you already have a project that uses the default name (LoyaltyAward), change the project name in the **New event project name** field.
8. Click **Finish**.

The imported project appears in the Event Explorer view.



The project contains two business objects, Customer details and Order totals, which are used in the averages calculation. An incoming event, Order processing complete, is included in the **Events** folder, along with the event object Order details. In the **Actions** folder, the project contains the action Send loyalty award, and an action object, Award details. This action is used to issue an award to customers who meet the loyalty criteria.

To run the completed tutorial:

1. Make sure that you are in the Event perspective. Click **Window > Open Perspective > Other > Event**.
2. In the Event Explorer, double-click event project items to view them in the default editors.
3. Deploy the **LoyaltyAward** project to the event run time. In the Event Explorer view, right-click the LoyaltyAwards project and then click **Deploy**.

 **Start this tutorial:** “Task 1: Creating an event project” on page 40.

Related information:

“Tutorial scenario” on page 38

The average of event values in a specified time period can be calculated in Decision Server Events. In this tutorial, you create a small project to calculate the average cost of purchases that a customer makes in a department store. The store uses the average to provide loyalty awards to a specific set of customers.

Task 1: Creating an event project

In this task, you create an event project and add an event and an event object. You also create an action and a corresponding action object.

When you complete this task, these event and action objects are added to the event project:

- The event object `Order details` is located in the **Events** category. This object defines the event that is generated each time a customer makes a purchase.
 - The action object `Award details` is located in the **Actions** category. This object defines the action that sends a loyalty award to a customer who, over the past five purchases, spent more than \$50 on average.
1. Create an event project to contain the tutorial application.
 - a. Right-click in the Event Explorer view, then click **New > Event Project**.
 - b. Enter the name of the project, which is `LoyaltyAwards`.
 - c. Click **Finish**.
 2. Create the `Order processing complete` event.
 - a. Right-click the **assets** folder, then click **New > Event**.
 - b. In the New Event wizard, select **Create a blank event**, then click **Next**.
 - c. Enter `Order processing complete` as the name of the event, then click **Finish**. The wizard closes and the event is opened in the Event Editor.
 - d. Right-click `LoyaltyAwards`, then click **Refresh**, to see the event you created in the Event Explorer view.
 3. Add the `Order details` event object.
 - a. In the Event Editor, in the Event Objects section, click **Add**. The **New event object** wizard is displayed.
 - b. Click **Add a new blank event object**, then click **Next**.
 - c. Enter `Order details` as the name of the object, then click **Finish**. The wizard closes and the event object is opened in the Event Object Editor.
 4. Add three fields to the `Order details` event object.
 - a. In the Event Object Editor, in the Fields section, click **Add**.
 - b. In the New Field wizard, name the first field `order date`, then select **DateTime** as the data type.
 - c. Click **Finish**.
 - d. Repeat the steps to add the `order amount` field with the data type **Real**, and the `Customer ID` field with the data type **String**.
 - e. Click **File > Save All**. The `Order details` event object includes the following fields:



Note: You can ignore the warning message in the Event Object Editor window because it is resolved during a later stage in the tutorial.

- f. Close the Event Object Editor and the Event Editor.
5. Create the Send loyalty award action.
 - a. Right-click the **assets** folder, then click **New > Action**.
 - b. In the New Action wizard, select **Create a blank action**, then click **Next**.
 - c. Enter Send loyalty award as the name of the action, then click **Finish**. The wizard closes and the action is opened in the Action Editor.
 - d. Right-click LoyaltyAwards, then click **Refresh**, to see the action you created in the Event Explorer view.
6. Add the Award details action object.
 - a. In the Action Editor, in the Action Objects section, click **Add**. The **New Action Object** wizard is displayed.
 - b. Click **Add a new blank action object**, then click **Next**.
 - c. Enter Award details as the name of the action object, then click **Finish**. The wizard closes and the event object is opened in the Action Object Editor.
7. Add two fields to the Award details action object.
 - a. In the Action Object Editor, in the Fields section, click **Add**.
 - b. In the New Field wizard, name the first field customer ID, then select **String** as the data type.
 - c. Click **Finish**.
 - d. Repeat the steps to add the award amount field with the data type **Real**.

Note: You can ignore the error message in the Action Object Editor window because it is resolved during a later stage in the tutorial.

- e. Click **File > Save All**.
- f. Close the Action Object Editor and the Action Editor.

In the Event Explorer view, the **LoyaltyAwards** project contains an event and an action with related objects, as shown in this example:



In the next task, you create and configure the business objects that contain the data used to complete the calculations.

 **Next:** “Task 2: Defining the business objects”

Task 2: Defining the business objects

In this task, you create and configure the business objects that contain the data used to complete the calculations. You also add field constructors to define how information in the event is passed to the business objects.

1. Create a new business object called `Order totals`. Configure this business object as an accumulating array.
 - a. Right-click the **assets** folder, then click **New > Business Object**.
 - b. Enter `Order totals` as the name of the business object, then click **Next**.
 - c. In the New Business Object window, select **Start with a blank business object**, then click **Finish**.
 - d. The `Order totals` object opens in the Business Object Editor.
 - e. Click to expand the Context-Scoped Settings section.
 - f. Select the option **Maintains an array of values from multiple events**.
 - g. Select the option **Specify the number of events over which to retain values in the array**.
 - h. Enter 5 next to **Maximum number of occurrences**.

This example shows the Context-Scoped Settings section with the correct options selected:

▼ Context-Scoped Settings

Configure this business object to store values received across multiple events within a context.

- Contains data received from a single event.
Specify how many instances of the business object exist at run time in response to the incoming event.

- Always treat as a single instance (scalar)
- Always treat as multiple instances (array)
- Can be either

- Maintains a single set of values based on a calculation of values from multiple events (summary instance).

- Maintains an array of values from multiple events (accumulating array).

- Specify the number of events over which to retain values in the array.

Maximum number of occurrences

- Specify the period of time over which to retain values in the array.

Days:

Hours:

Minutes:

Seconds:


- Keep the values in this object persistent across system shutdowns.

- i. Click **File > Save**.
2. Define two business object fields in the Order totals business object: one field named order amount to contain the transaction value and one field named order date to contain the transaction date.
 - a. In the Fields section, click **Add**.
 - b. In the New Field window, enter order amount as the first new field name, and select **Real** as the data type.
 - c. Click **Finish**.
 - d. Repeat the steps to add order date as the second field and **DateTime** as the data type.
 - e. Click **Finish**. This example shows the Order totals business object with the correct fields added:

 **Business object: Order totals** 1 warning detected

▼ **General Information**

Name: Order totals

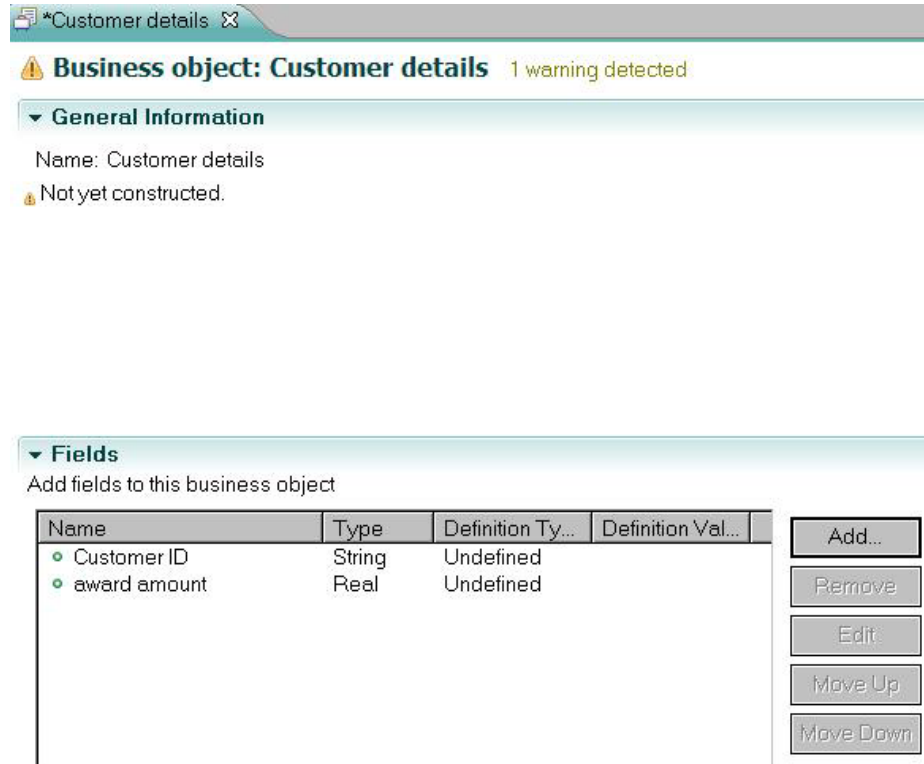
 Not yet constructed.

▼ **Fields**

Add fields to this business object

Name	Type	Definition Ty...	Definition Val...	
<input type="radio"/> order amount	Real	Undefined		<input type="button" value="Add..."/>
<input type="radio"/> order date	DateTi...	Undefined		<input type="button" value="Remove"/>
				<input type="button" value="Edit"/>
				<input type="button" value="Move Up"/>
				<input type="button" value="Move Down"/>

- f. Click **File > Save**.
3. Create a second business object to hold the customer-specific details, and define two business object fields, one to contain the context ID and the other to contain the loyalty details.
 - a. Create a new business object called Customer details.
 - b. Using the Business Object Editor, define business object fields called Customer ID and award amount in the Customer details business object. This example shows the Customer details object with the correct fields added:



- c. Click **File > Save**.
- d. Close the Business Object Editor.
4. Verbalizations are created automatically for business objects and business object fields. Edit the default verbalizations to improve the phrasing for these assets that you use later in the tutorial to create filters and events.
 - a. Open the Order totals object in the Business Object Editor.
 - b. To improve the phrasing of the verbalization, change the default verbalization of the business object. Click **Edit** to modify the verbalization term, and change the term from order totals to simply order.
 - c. In the Fields table, double-click **order amount** to edit the field. The order amount field details are displayed in the Field window.
 - d. In the Verbalization section of the window, verify that the default verbalization has been added to the field. The default verbalization is "order amount of this." This verbalization creates awkward English phrasing. The phrasing is clearer if it reads simply "amount." In the next step, you change the template used to build the verbalization, which improves the phrasing.
 - e. Click **Edit** to modify the verbalization term, and change the term from order amount to amount.
 - f. In the Template field, change the text from {amount} of {this} to {this} {amount}. The verbalization phrase is changed to "an order amount".
 - g. Repeat the steps to change the default verbalization of the order date business object field to date.
 - h. Click **File > Save**.
 - i. Close the Business Object Editor.
5. Add three field constructors to define how the business object fields in the two business objects are populated. Field constructors define the mapping from the event object to the business object. The field constructors that you define have a

definition type of **Field**. A definition type of field means that the mapping moves data from a field in the event object to a field in the business object. Therefore, each field receives its value from the corresponding event object field. The field constructors that you define are named order date, order amount, and Customer ID.

- a. In the Event Explorer view, click **assets > Events**, then double-click **Order processing complete** to open the event in the editor.
- b. In the Event Objects section, double-click the **Order details** event object to open it in the Event Object Editor.
- c. In the Field Constructors section at the bottom of the editor view, click **Add** on the right side of the business object table.
- d. In the Field Constructor window, expand **Customer details** and **Order totals**. Select **order date**, **order amount**, and **Customer ID**. Click **Finish**.
- e. In the Field Constructors table, double-click the **Order totals > order date** row to map this business object field to the event object.
- f. In the Field Constructor window, click **Definition type** in the Definition section, and select **Field**.
- g. In the **Event object field** list, select **order date**.
- h. Click the **Overview** tab to return to the Field Constructors section.
- i. Set the event object fields for the following business objects in the Field Constructors table:
 - Edit **Order totals > order amount** and set the field to **order amount**.
 - Edit **Customer details > Customer ID** and set the field to **customer ID**.

This example shows the Order details object with the correct field constructors added:

▼ **Field Constructors**
Add business objects that will receive data from the fields in this event object:

Business Object	Field	Data Type	Definition Type	Definition Value
Customer details	Customer ID	String	Field	Customer ID
Order totals	order amount	Real	Field	order amount
Order totals	order date	DateTime	Field	order date

- j. Click **File > Save**.
 - k. Close the Event Object Editor and the Event Editor.
6. Add business object definitions to the action object fields. Action object fields must receive values from the business objects so that the required customer data is added to the action. If an action object field does not include a field definition, then the value of the field is null.
- a. Under the **assets** folder, select the **Actions** folder and click to expand the **Send loyalty award** action.
 - b. Double-click the **Award details** action object to open it in the Action Object Editor.
 - c. In the Fields section, double-click **customer ID** to edit the field.
 - d. In the Definition section, select **Field** from the Definition Type list.
 - e. Select **Customer details** as the business object and **Customer ID** as the business object field. As a result of the defined relationship between the business object and the action object, the value from the selected business object is passed directly to the action field when the tutorial application is deployed.

- f. Repeat the steps to define a business object definition for the award amount field. Select **Customer details** as the business object and **award amount** as the business object field.
- g. Click **File > Save**.
- h. Close the Action Object Editor and the Action Editor. This example shows the expanded **LoyaltyAwards** event project in the Event Explorer view, at this stage of the tutorial.



In the next task, you define the event rule and filters to enable the averages calculation.

Next: “Task 3: Defining the event rule and filters”

Related concepts:

Field constructors

Related tasks:

Defining business objects

Defining business object fields

Task 3: Defining the event rule and filters

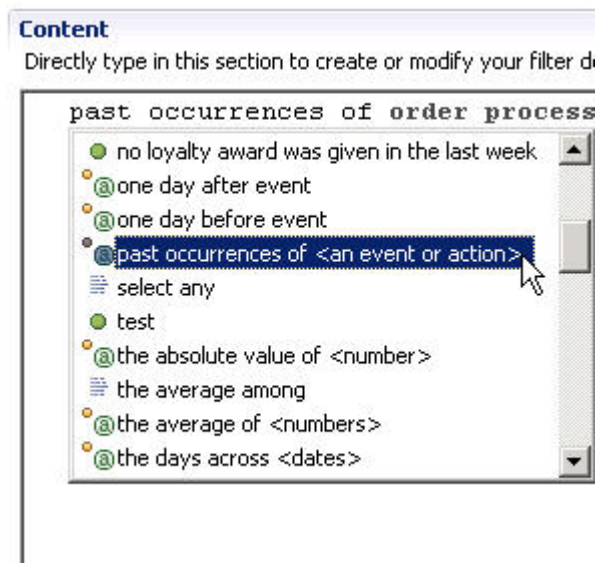
In this task, you create the event rule and the filters that complete the averages calculation and determine when to send the loyalty award.

You create filters to calculate the average value of purchases, determine the time period in which the purchase were made, and check to make sure that the customer did not receive a loyalty award recently, as described in the table. You also create a rule called Check for loyalty award status which uses the filters.

Filter name	Filter function
the average purchase amount is greater than 50	Calculates the average value of the last 5 purchases made by a customer.
more than 4 orders placed in the last 2 weeks	Determines whether the last five purchases made by a customer were within a 2 week period.
no loyalty award was given in the last week	Confirms that the customer did not receive a loyalty award within the last week.

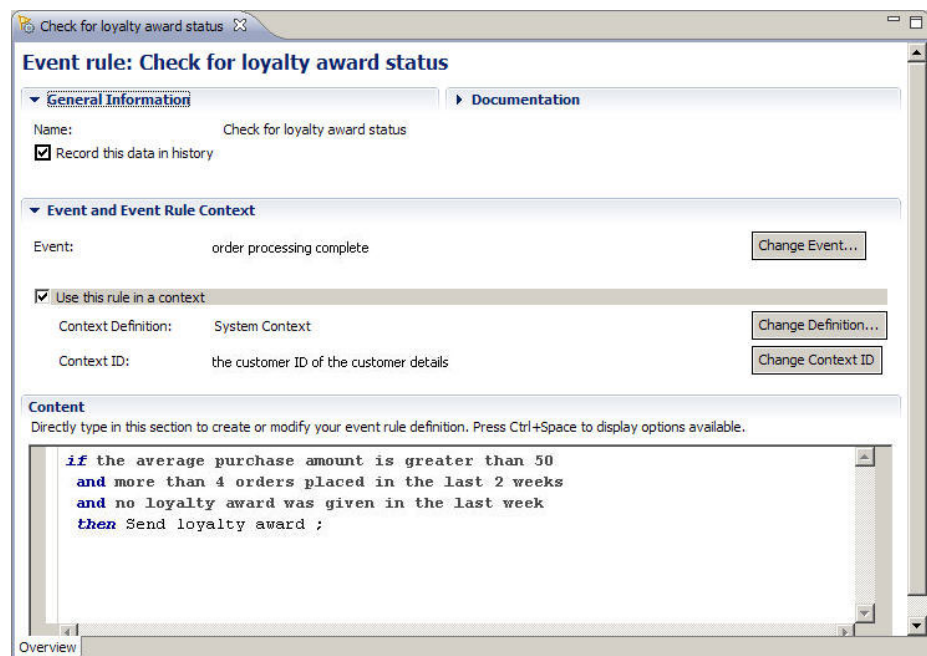
1. Create three filters as described in the table.

- a. Right-click the **assets** folder, then click **New > Filter**.
- b. Enter the average purchase amount is greater than 50 as the name of the filter, then click **Finish**. The wizard closes and the filter is opened in the Filter Editor.
- c. In the Filter Editor window, click to place the edit cursor inside the **Content** field.
- d. Press Ctrl+Spacebar to display the Content Assist box.
- e. Scroll down in the Content Assist box until you see the phrase **the average of <numbers>**.
- f. Double-click to select **the average of <numbers>** from the list and add this phrase to the filter content field.
- g. Continue editing the filter content and add these phrases:
 - **the orders**
 - **amounts**
 - **is more than**
 - **<number>**
- h. Click the variable **<number>** and enter 50 to replace the variable with the value 50. This example shows the completed filter content:
the average of the orders amounts is more than 50
- i. Click **File > Save**.
- j. Repeat the steps to create two additional filters. Use the Content Assist box to locate and add the filter content.



- Create a filter named **more than 4 orders placed in the last 2 weeks** with this content:
past occurrences of order processing complete within 2 weeks is at least 4
 - Create a filter named **no loyalty award was given in the last week** with this content:
past occurrences of send loyalty award within 1 weeks equals 0
- k. Close the Filter Editor.
 2. Create the event rule that uses the filters.
 - a. Right-click the **assets** folder, then click **New > Event Rule**.

- b. In the New Event Rule wizard, enter Check for loyalty award status as the name of the event rule.
- c. Click **Next**.
- d. In the Select Event panel, click **order processing complete** to indicate that the event rule applies to the selected event.
- e. Click **Next**.
- f. In the Select the context panel, click **System Context**, then click **Next**.
- g. In the Define the context ID panel, click **the customer ID of the customer details**. You must set the context ID in the event rule because the event rule references an accumulating array object and because the rule also checks for past occurrences of events and actions.
- h. Click **Finish**. The event rule is opened in the Rule Editor.
- i. To add the event rule content, click inside the **Content** field to place the edit cursor in the field, then press Ctrl+Spacebar to open the Content Assist box.
- j. Select **if <conditions>** from the list. Continue editing the rule content and add these phrases:
 - **the average purchase amount is greater than 50.**
 - **and <condition>.**
 - **more than 4 orders placed in the last 2 weeks.**
 - **and <condition>.**
 - **no loyalty award was given in the last week.**
 - **then <actions>.**
 - **send loyalty award.**
- k. Click the semicolon symbol (;), which indicates the end of the rule content. This example shows the completed event rule:



3. Save the event rule and close the Event Editor.

In the next task, you deploy the tutorial application.

Next: “Task 4: Deploying the application” on page 50

Task 4: Deploying the application

In this task, you deploy the application you have built.

Now that you have completed the event project which defines the tutorial application, you must deploy the event project to the event run time which is running on your system so that you can confirm that the application works.

1. In the Event Explorer view, right-click the **LoyaltyAwards** project and then click **Deploy**.
2. In the Deploy window, check to make sure that the default option, **Deploy all assets**, is selected, then click **Next**.
3. Select an existing event run time from the Deploy window, then click **Finish**. The event project assets are deployed to the event run time.

If the deployment fails, an error message is displayed in the **Problems** view. Check that you have defined the assets in the event project as described in the previous tutorial tasks.

You have completed the **Calculating averages to identify event patterns** tutorial, and learned how to create and configure an event project with an accumulating array business object.

Tutorial: Comparing an event with a previous event

This tutorial describes comparing an event that Decision Server Events recently received with the previous occurrence of the same event.

You can identify a pattern of business events by comparing an event that just occurred (the event that Decision Server Events receives) with a previous occurrence of the same event.

Before you start

In this tutorial, you create an event project to compare an event that just happened (the event that Decision Server Events receives) with an event that happened previously (the previous occurrence of the same event). The compared events are a current meter reading and a previous meter reading.

When a smart electricity meter reading is received, it is compared to the previous meter reading. If the current reading is higher than the previous reading, the usage cost is calculated and a customer bill is prepared, but if the current reading is lower than the previous reading, an alert is sent to the utility company to determine if the meter is faulty, or has been tampered with. This topic describes the prerequisites for running the tutorial.

To complete the tutorial, you complete these tasks:

- “Task 1: Creating an event project” on page 53
- “Task 2: Defining the business objects” on page 55
- “Task 3: Defining the event rule and filter” on page 60
- “Task 4: Deploying the application” on page 62

What you will learn

In this tutorial, you will learn these skills:

- Use Decision Server Events to compare a current event with a previous event.

- Configure a business object as a summary instance object to collect and store transaction data.
- Use JavaScript in a field constructor to copy transaction data from a previous transaction into the summary object.
- Perform a specific action based on the results of comparing the previous and current events.

This tutorial takes approximately 2 hours to complete.

What you need to know

This tutorial is intended for developers who want to learn how to develop event projects in Decision Server Events. The procedures in the tutorial assume that you are familiar with the Eclipse environment, and that you have completed the Getting started with event rules tutorial.

 **Start this tutorial:** “Task 1: Creating an event project” on page 53.

Tutorial scenario

You can identify a pattern of business events by comparing an event that just occurred (the event that Decision Server Events receives) with a previous occurrence of the same event. For example, an electricity company might compare a current meter reading with the previous meter reading for the same account to determine whether the meter is functioning correctly.

An electricity utility company uses smart meters to automatically send electricity usage readings back to the company. The utility company is using Decision Server Events, so each smart meter reading is received as an event. When a current meter reading is received it is compared to the previous meter reading and if the current reading is higher than the previous reading, the usage cost is calculated and a customer bill is prepared. If the current reading is lower than the previous reading, an alert is sent to indicate that there might be a problem with the meter. The utility company conducts an investigation to determine whether the meter is faulty or whether someone has tampered with the meter.

To complete the tutorial, you complete these tasks:

1. Create an event project and add a meter reading event and an action to initiate an investigation if the event data meets specific criteria.
2. Create business objects to hold transaction summary data and customer-specific information.
3. Create a filter to compare event values.
4. Define the event rule that uses the value comparison to trigger an action.
5. Deploy and test the application created by the completed tutorial.

To compare an event with a previous event in Decision Server Events, you must define a summary instance business object. This object uses the **Keep the values in this type of object persistent across system shutdowns** option because the application must preserve the previous meter readings to perform the comparison function. The meter readings are compared each time the smart meter sends a current reading. The event project contains an additional business object that tracks the amount of electricity used by a specific customer, therefore a unique customer reference field is used as the context ID for the business object.

 **Start this tutorial:** “Task 1: Creating an event project” on page 53.

Related information:

“Running the completed tutorial”

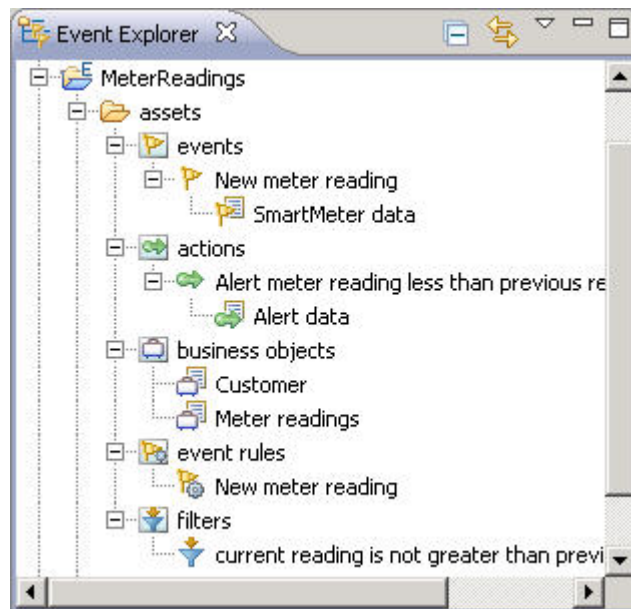
You can import and view the tutorial in its completed state. When completed, the tutorial contains an event that you can run to compare an event that just happened with an occurrence of the same event that happened previously.

Running the completed tutorial

You can import and view the tutorial in its completed state. When completed, the tutorial contains an event that you can run to compare an event that just happened with an occurrence of the same event that happened previously.

To import the completed project and run the tutorial application:


1. Download the completed tutorial project for comparing events. Right-click `MeterReadings.xml` and save the file to your local system.
2. In the Event Explorer view, right-click and select **Import**.
3. In the Import window, click **Event Designer > Event Project from XML File**, then click **Next**.
4. Click **Browse** to select the event project file to import: `MeterReadings.xml`.
5. Navigate to the location of the `MeterReadings.xml` file, select the file, then click **Open**, then click **Next**.
6. Click **Next** in the Project Assets panel.
7. Optional: If you already have a project that uses the default project name (`MeterReadings`), change the name in the **New event project name** field.
8. Click **Finish**. The imported project appears in the Event Explorer view:



The project contains an event named `New meter reading` and an associated event object named `SmartMeter data`. Two business objects, `Customer` and `Meter readings`, contain the summary data and the unique customer ID. In the **Actions** folder, the project contains the action `Alert meter reading less than previous reading`, which prompts the utility company to conduct an investigation to determine whether the meter is faulty.

To run the completed tutorial:

1. Make sure that you are in the Event perspective. Click **Window > Open Perspective > Other > Event**.
2. In the Event Explorer, double-click event project items to view them in the default editors.
3. Deploy the MeterReadings project to the event run time. In the Event Explorer view, right-click the MeterReadings project and then click **Deploy**.

 **Start this tutorial:** “Task 1: Creating an event project.”

Related information:

“Tutorial scenario” on page 51

You can identify a pattern of business events by comparing an event that just occurred (the event that Decision Server Events receives) with a previous occurrence of the same event. For example, an electricity company might compare a current meter reading with the previous meter reading for the same account to determine whether the meter is functioning correctly.

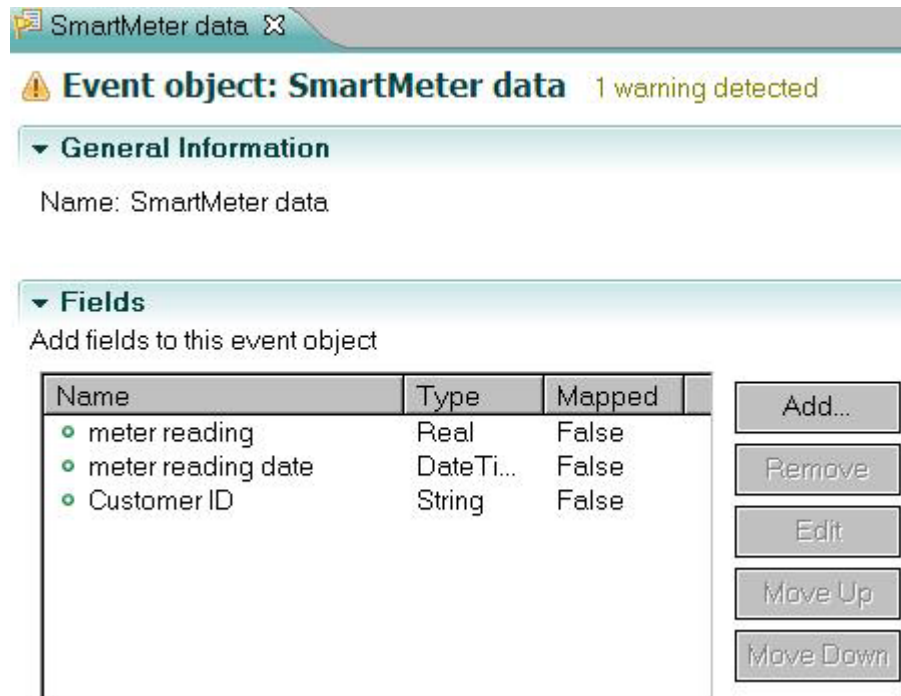
Task 1: Creating an event project

In this task, you create an event project to contain the comparing events tutorial application. You also add the New meter reading event, an event object, an action, and a corresponding action object. Define the event fields to contain details of the current transaction, the date of the transaction, and the context ID.

When you complete this task, the following event and action objects are added to the event project:

- The event New meter reading is located in the **Events** category. The Smart meter data event object contains the event fields.
 - The action Alert meter reading less than previous reading is located in the **Actions** category. The Alert data action object contains the event fields.
1. Create an event project to contain the tutorial application.
 - a. Right-click in the Event Explorer view, then click **New > Event Project**.
 - b. Enter the name of the project: MeterReadings.
 - c. Click **Finish**.
 2. Create the New meter reading event.
 - a. Right-click the **assets** folder, then click **New > Event**.
 - b. In the New Event wizard, select **Create a blank event**, then click **Next**.
 - c. Enter New meter reading as the name of the event , then click **Finish**. The wizard closes and the event is opened in the Event Editor.
 3. Create the SmartMeter data event object.
 - a. In the Event Editor, in the Event Objects section, click **Add**. The **New event object** wizard is displayed.
 - b. Click **Next**.
 - c. Enter SmartMeter data as the name of the event object, then click **Finish**.
 4. Add three fields to the SmartMeter data event object.
 - a. In the Event Object Editor, in the Fields section, click **Add**.
 - b. In the New Fields wizard, name the first field meter reading, then select **Real** as the data type.
 - c. In the New Fields wizard, click **Finish**.

- d. Repeat the steps to add the meter reading date field with the data type **DateTime**, and the Customer ID field with the data type **String**.
- e. Click **File > Save All**. The SmartMeter data event object includes the added fields, as shown:

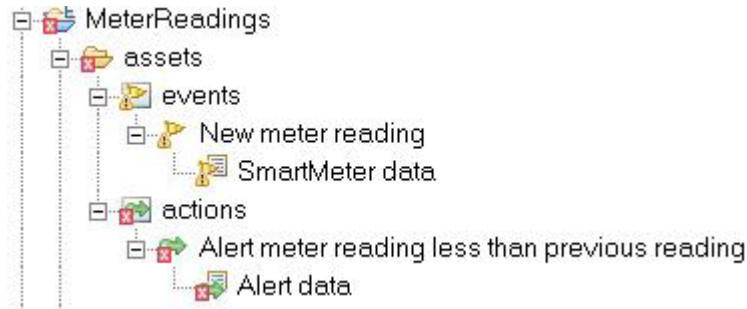


- f. Close the Event Object Editor and the Event Editor.
5. Create the Alert meter reading less than previous reading action.
 - a. Right-click the **assets** folder, then click **New > Action**.
 - b. In the New Action wizard, select **Create a blank action**, then click **Next**.
 - c. Enter Alert meter reading less than previous reading as the name of the action, then click **Finish**. The wizard closes and the action is opened in the Action Editor.
6. Add the Alert data action object.
 - a. In the Action Editor, in the Action Objects section, click **Add**. The **New Action Object** wizard is displayed.
 - b. Make sure that the option to **Add a new blank action object** is selected, then click **Next**.
 - c. Enter Alert data as the name of the action object, then click **Finish**.
7. Add five fields to the Alert data action object: Customer ID, current reading, previous reading, current reading date, and previous reading date.
 - a. In the Action Object Editor, in the Fields section, click **Add**.
 - b. In the New Field wizard, name the first field Customer ID, then select **String** as the field type.
 - c. In the New Field wizard, click **Finish**.
 - d. Repeat the steps to add the following fields and field types:
 - current reading, field type **Real**
 - previous reading, field type **Real**
 - current reading date, field type **DateTime**
 - previous reading date, field type **DateTime**

Note: You can ignore the error message in the Action Object Editor window because it is resolved during a later stage in the tutorial.

- e. Click **File > Save**.
- f. Close the Action Object Editor and the Action Editor.
- g. Click **File > Save All**.

In the Event Explorer view, the **MeterReadings** project contains an event and an action, and objects that work with them, as shown in this example:



In the next task, you create the business objects that contain the summary data and the context ID.

 **Next:** “Task 2: Defining the business objects”

Task 2: Defining the business objects

In this task, you create a business object to hold the summary data, and a second business object to hold customer-specific information, such as the customer ID. The first business object must be configured as a summary instance object.

1. Create a new business object named Meter readings. Configure this business object as a summary instance.
 - a. Right-click the **assets** folder, then click **New > Business Object**.
 - b. Enter Meter readings as the name of the business object, then click **Next**.
 - c. In the New Business Object window, select **Start with a blank business object**, then click **Finish**. The Meter readings business object opens in the Business Object Editor.
 - d. Expand the Context-Scoped Settings section, then select the option **Maintains a single set of values based on a calculation of values from multiple events**.
 - e. In the same section, select the option **Keep the values in this object persistent across system shutdowns** option. The following example shows the Context-Scoped Settings section with the correct options selected:

▼ Context-Scoped Settings

Configure this business object to store values received across multiple events within a context.

- Contains data received from a single event.
 - Specify how many instances of the business object exist at run time in response to the incoming event.
 - Always treat as a single instance (scalar)
 - Always treat as multiple instances (array)
 - Can be either
- Maintains a single set of values based on a calculation of values from multiple events (summary instance).
- Maintains an array of values from multiple events (accumulating array).
 - Specify the number of events over which to retain values in the array.

Maximum number of occurrences
 - Specify the period of time over which to retain values in the array.

Days:

Hours:

Minutes:

Seconds:
- Keep the values in this object persistent across system shutdowns.

- f. Click **File > Save**.
2. Using the Business Object Editor, define four business object fields in the Meter readings summary instance object: one field to contain the previous transaction data, one field to contain the latest transaction data, one field to contain the previous transaction date, and one field to contain the latest transaction date.
 - a. In the Fields section, click **Add**.
 - b. In the New Field window, enter current reading as the first new field name, and select **Real** as the type.
 - c. Click **Finish**.
 - d. Repeat the steps to add these fields and field types:
 - previous reading, field type **Real**
 - current reading date, field type **DateTime**
 - previous reading date, field type **DateTime**
 - e. Click **File > Save**. The following example shows the Meter readings business object with the correct fields added:

Business object: Meter readings 1 warning detected

General Information

Name: Meter readings

⚠ Not yet constructed.

Fields

Add fields to this business object

Name	Type	Definition Ty...	Definition Val...
<input type="radio"/> current reading	Real	Undefined	
<input type="radio"/> previous reading	Real	Undefined	
<input type="radio"/> current reading date	DateTi...	Undefined	
<input type="radio"/> previous reading date	DateTi...	Undefined	

Add...
Remove
Edit

3. Create the Customer business object to hold the customer-specific information.
 - a. Create a new business object named **Customer**, as described in the previous steps.
 - b. Using the Business Object Editor, define a business object field called Customer ID. The following example shows the Customer business object with the correct field added:

*Customer

Business object: Customer 1 warning detected

General Information

Name: Customer

⚠ Not yet constructed.

Fields

Add fields to this business object

Name	Type	Definition Ty...	Definition Val...
<input checked="" type="radio"/> Customer ID	String	Undefined	

Add...
Remove

- c. Click **File > Save All**.
4. Verbalizations are created automatically for business objects and business object fields. Edit the default verbalizations to improve the phrasing for these assets that you use later in the tutorial to create filters and events.
 - a. In the Business Object Editor, open the Meter readings object.
 - b. In the Fields section, double-click **current reading** to edit the field. The current reading field properties are displayed in the Field window.

- c. In the Verbalization section of the window, verify that the default verbalization {current reading} of {this}, has been added to the field properties.
 - d. Click the **Overview** tab to return to the business object Fields section.
 - e. Repeat the steps to verify the default verbalizations for the remaining fields in the Meter readings business object.
 - f. Click the **Overview** tab to return to the Meter readings business object panel, and verify that the default verbalization of the business object is "meter readings." This verbalization creates awkward English phrasing. The phrasing is clearer if it reads simply "meter." In the next step, you change the template used to build the verbalization, which improves the phrasing.
 - g. Click **Edit** next to the verbalization term to simplify the verbalization from "meter readings" to "meter."
 - h. Click **File > Save**.
 - i. Close the Business Object Editor window.
5. Add four field constructors to the SmartMeter data event object to define how the business object fields in the two business objects are populated. Field constructors define the mapping from the event object to the business object. The mapping moves data from a field in the event object to a field in the business object so that each field receives its value from the corresponding event object field. The field constructor for the latest transaction data uses JavaScript to copy the data from the previous transaction data field in the summary object before the new current value is copied.
 - a. In the Event Explorer view, click **assets > Events > New meter reading** to open the event.
 - b. Double-click the SmartMeter data event object to open it in the Event Object Editor.
 - c. In the Field Constructors section at the bottom of the editor view, beside the business object table, click **Add**.
 - d. In the Field Constructor window, click **Select all**, then click **Finish**.
 - e. In the Field Constructors table, double-click the **Meter readings > previous reading** row to map this business object field to the event object.
 - f. In the Field Constructor window, click **Definition Type** in the Definition section, and select **Javascript**.
 - g. In the **Javascript expression** field, enter this script:


```
if (Meter readings.current reading == null)
  {0}
else
  {Meter readings.current reading}
```

The **previous reading** field constructor uses JavaScript to copy the data from the current reading field in the summary object before the current value is copied. If the value of current reading in the Meter readings business object is null, then the value of previous reading is set to zero. Otherwise, previous reading is set to the value of current reading from the business object.
 - h. Click the **Overview** tab to return to the Field constructors section.
 - i. Double-click the **Meter readings > previous reading date** row in the Field constructors table.
 - j. In the Field Constructor window, click **Definition type** in the Definitions section, and select **Javascript**.
 - k. In the **Javascript expression** field, enter this script:

Meter readings.current reading date

The fields current reading and current reading date are populated by copying the data from the event fields into the fields in the summary object. The value of previous reading date is populated by copying the value of current reading date from the business object.

- l. Click the **Overview** tab to return to the Field constructors section.
- m. Double-click the **Meter readings > current reading** row in the Field constructors table.
- n. In the Field Constructor window, click **Definition type** in the Definitions section, and select **Field**.
- o. In the **Field name** list, select **meter reading**.
- p. Click the **Overview** tab to return to the Field constructors section.
- q. Create additional field constructors for these fields:
 - Double-click **Meter readings > current reading date**, then select **meter reading date** as the **Field name**.
 - Double-click **Customer > Customer ID**, then select **Customer ID** as the **Field name**.

The following example shows the SmartMeter data event object with the correct field constructors added:

▼ **Field Constructors**
Add business objects that will receive data from the fields in this event object:

Business Object	Field	Data Type	Definition Type	Definition Value
Customer	Customer ID	String	Field	Customer ID
Meter readings	current reading	Real	Field	meter reading
Meter readings	previous reading	Real	Javascript	if (Meter readings.curre...
Meter readings	current reading date	DateTime	Field	meter reading date
Meter readings	previous reading date	DateTime	Javascript	Meter readings.current r...

- r. Click **File > Save**.
 - s. Close the Event Object Editor.
6. Add business object definitions to the action object fields. Action object fields must receive values from the business objects so that the required customer data is added to the action. If an action object field does not include a field definition, then the value of the field is null.
- a. Under the **assets** folder, click **Actions > Alert meter reading less than previous reading**.
 - b. Double-click the **Alert data** action object to open it in the Action Object Editor.
 - c. In the Fields section, double-click **Customer ID** to edit the field.
 - d. In the Definition section, select **Field** from the Definition Type list.
 - e. Select **Customer** as the business object and **Customer ID** as the business object field.
 - f. Repeat the steps to add definitions for the current reading, previous reading, current reading date, and previous reading date fields. As a result of the defined relationship between the business objects and the action object, the value from the selected business objects is passed directly to the action field when the tutorial application is deployed.

The following examples shows the completed business object definitions:

Fields

The fields in the action are grouped into action objects, which can be optionally reused by other actions.

Name	Type	Definition Type	Definition Value
Customer ID	String	Field	Customer ID
current reading	Real	Field	current reading
previous reading	Real	Field	previous reading
current reading date	DateTime	Field	current reading date
previous reading date	DateTime	Field	previous reading date

Buttons: Add..., Remove, Edit, Move Up

- g. Click **File > Save**.
- h. Close the Action Object Editor.

The following example shows the current MeterReadings project in the Event Explorer view:



In the next task, you define the event rule that compares the latest and previous event values.

Next: “Task 3: Defining the event rule and filter”

Related concepts:

Field constructors

Related tasks:

Defining business objects

Defining business object fields

Task 3: Defining the event rule and filter

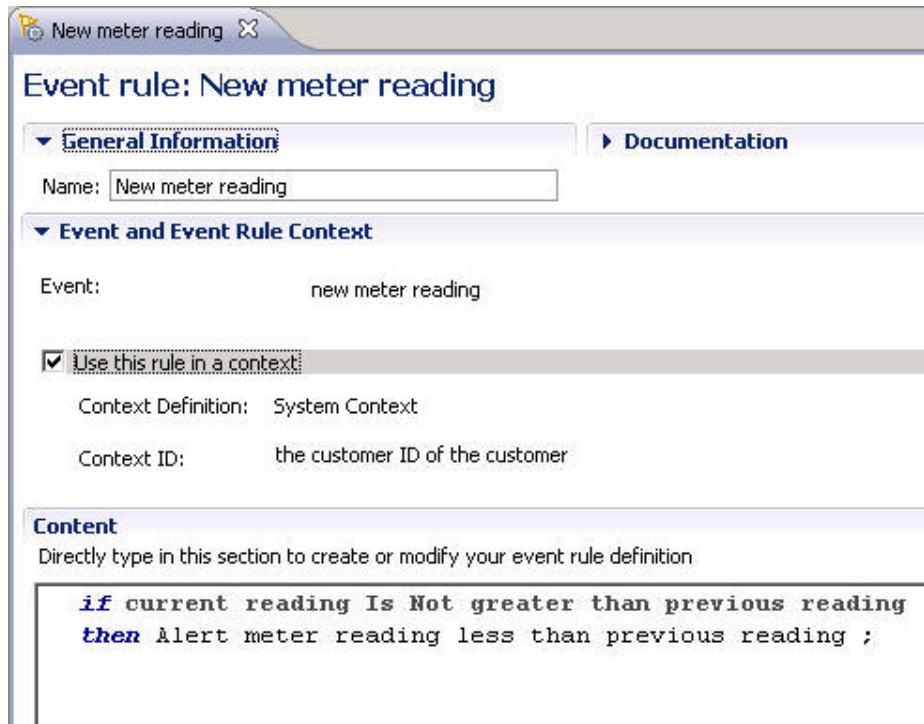
In this task, you define the event rule that compares the latest and previous event values. You also create a filter to compare meter reading values and determine if the latest reading is greater than, or less than, the previous reading.

If the results of comparing the data from the latest event and the previous event are unusual, for example, if the latest meter reading is less than the previous meter reading, the event rule triggers an action that prompts the utility company to investigate.

1. Create the filter to compare the latest event value with the previous event value.
 - a. Right-click the **assets** folder, then click **New > Filter**.

- b. Enter current reading is not greater than previous reading as the name of the filter, then click **Finish**. The wizard closes and the action is opened in the Filter Editor.
 - c. In the Filter Editor window, click to place the edit cursor inside the **Content** field.
 - d. Press Ctrl+Spacebar to display the Content Assist box.
 - e. Scroll down in the Content Assist box until you see the phrase **the current reading of <a meter>**.
 - f. Double-click to select **the current reading of <a meter>** from the list, and add this phrase to the filter content field.
 - g. Continue editing the filter content and add these phrases:
 - **the meter**
 - **is less than <a number>**
 - **the previous reading of <a meter>**
 - **the meter**

This example shows the completed filter content:
the current reading of the meter is less than the previous reading of the meter
 - h. Click **File > Save**, then close the Filter Editor.
2. Create the event rule that uses the filter.
 - a. Right-click the **assets** folder, then click **New > Event Rule**.
 - b. In the New Event Rule wizard, enter New meter reading as the name of the event rule.
 - c. Click **Next**.
 - d. In the Select Event panel, click **new meter reading** to indicate that the new event rule applies to the selected event.
 - e. Click **Next**.
 - f. In the Select context panel, click **System context**, then click **Next**.
 - g. In the Define the context ID panel, select **the customer ID of the customer** as the context so that the event rule can use the customer ID to reference the summary object.
 - h. Click **Finish**. The event rule is opened in the Rule Editor.
 - i. Add the event rule content.
 - 1) Click inside the **Content** field to place the edit cursor in the field.
 - 2) Press Ctrl+Spacebar to open the Content Assist box.
 - 3) Select **if <conditions>** from the list.
 - j. Continue editing the rule content and add these phrases:
 - **current reading is not greater than the previous reading**
 - **then <actions>**
 - **alert meter reading less than previous reading**
 - k. Click the semicolon symbol (;), which indicates the end of the rule content. This example shows the completed rule event:



3. Save the event rule and close the Event Editor.

In the next task, you deploy and test the tutorial application.

 **Next:** “Task 4: Deploying the application”

Task 4: Deploying the application

In this task, you deploy the tutorial application.

Now that you have completed the event project that defines the tutorial application, you must deploy the event project to the event run time that is running on your system so that you can confirm that the application works.

1. In the Event Explorer view, right-click the **MeterReadings** project name and then click **Deploy**.
2. In the Deploy window, check to make sure that the default option **Deploy all assets** is selected, then click **Next**.
3. Select an existing event run time from the Deploy window, then click **Finish**. The event project assets are deployed to the event run time.

If the deployment fails, an error message is displayed in the Problems view. Check that you have defined the assets in the event project as described in the previous tutorial tasks.

You have completed the tutorial. Use the Event Tester widget to test the behavior of the sample event project by submitting events.

Related tasks:

Testing event logic using the Event Tester widget

Tutorial: Tracking the state of something

This tutorial shows you how to track the state of something, such as the central heating system in a smart home, using events in Decision Server Events.

Learning objectives

In the tutorial scenario, the owner of a smart home uses events to record and track the state of the central heating furnace, such as starting, running, shutting down, or stopped.

Before you start

In this tutorial, you create an event project to track the state of an advanced central heating system in a home. In this scenario, as the owner of the home, you have installed smart house technology to control the central heating, and you use Decision Server Events to track events that capture the state of the system: starting, running, shutting down, or stopped.

To monitor the central heating status using Decision Server Events you must define a summary instance object. The fields in the business object are used to track various system states. You define filters and an event rule to compare the order of the furnace events within a specific time period. If an unexpected pattern of events is detected, the sequence of events might indicate a problem with the furnace. If a problem is detected, the DateTime fields in the business objects are passed as data to the event project actions triggered by the events.

To complete the tutorial, you complete these tasks:

- “Task 1: Creating an event project” on page 66
- “Task 2: Defining the business objects” on page 70
- “Task 3: Defining the event rules and filter” on page 75
- “Task 4: Deploying the application” on page 77

What you will learn

In this tutorial, you will learn these skills:

- Use Decision Server Events to track the state of a smart home heating system, by using events.
- Configure a business object as a summary instance object to collect and store data about the state of the furnace.
- Define field constructors to populate the business object fields.
- Create actions that are triggered if the events are not received in the expected order, or at the specified time.
- Use filters to check whether an event that corresponds to a specific furnace operation state is received within a specified time.

This tutorial takes about 2 hours to complete.

What you need to know

This tutorial is intended for developers who want to learn how to develop event projects in Decision Server Events. The procedures in the tutorial assume that you are familiar with the Eclipse environment, and that you have completed the Getting started with event rules tutorial.

 **Start this tutorial:** “Task 1: Creating an event project” on page 66.

Tutorial scenario

You can track the events related to a central heating system in a smart home by using Decision Server Events. In this tutorial scenario, the owner of a smart home uses events to record and track the state of the central heating furnace, such as starting, running, shutting down, or stopped.

Smart home technology controls various systems in a house such as the lights and the central heating. The smart home owner can use this technology to identify a possible burglary if a window is open at an odd time of day, or prevent water pipes from freezing by automatically managing the central heating controls when the weather forecast predicts cold temperatures. Using Decision Server Events, the smart home owner can track the state of a window (open or closed), or the central heating (on or off), and then respond to those states; for example, by alerting the police, or starting the furnace.

To complete the tutorial, you complete these tasks:

1. Create an event project and add events that capture the state of the furnace, along with actions that are performed as a result of the events.
2. Create business objects to hold the furnace status data.
3. Create a filter to track the timing of the furnace events.
4. Define the event rule that uses the furnace event timing or event sequence to trigger an action.
5. Deploy the application created by the completed tutorial.

 **Start this tutorial:** “Task 1: Creating an event project” on page 66.

Related information:

“Viewing the completed tutorial”

You can import and view the tutorial in its completed state. When completed, the tutorial contains an event that you can run to track the current state of the central heating system in a smart home.

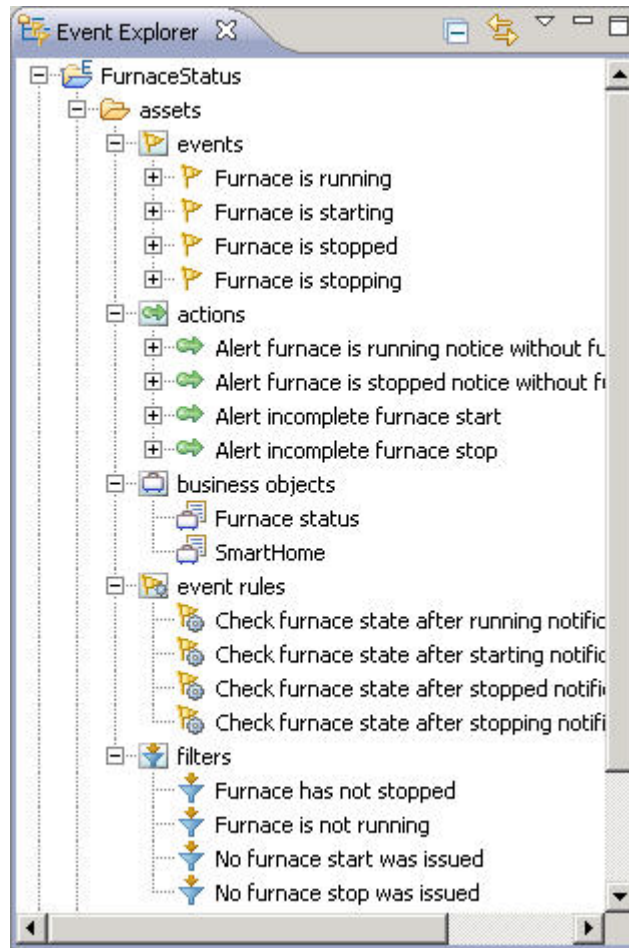
Viewing the completed tutorial

You can import and view the tutorial in its completed state. When completed, the tutorial contains an event that you can run to track the current state of the central heating system in a smart home.

To import the completed project and run the tutorial application:

1. Download the completed tutorial project for tracking the system state. Right-click `FurnaceStatus.xml` and save the file to your local system.
2. In the Event Explorer view, right-click and select **Import**.
3. In the Import window, click **Event Designer > Event Project from XML File**, then click **Next**.


4. Click **Browse** to select the event project file to import: FurnaceStatus.xml.
5. Navigate to the location of the FurnaceStatus.xml file, select the file, click **Open**, then click **Next**.
6. Click **Next** in the project assets panel.
7. In the project name panel, If you already have a project that uses the default project name, (FurnaceStatus), you can change the name in the **New event project name** field.
8. Click **Finish**. The imported project appears in the Event Explorer view.



The project contains four events that represent the status of the furnace, and an associated event object for each event. Two business objects, Furnace status and SmartHome, contain the furnace state summary data and the unique furnace ID. In the **Actions** folder, the project contains four alert actions that are triggered if the timing or sequence of the furnace events indicates a problem with the heating system.

To run the completed tutorial:

1. Make sure that you are in the Event perspective. Click **Window > Open Perspective > Other > Event**.
2. In the Event Explorer, double-click event project items to view them in the default editors.
3. Deploy the FurnaceStatus project to the event run time. In the Event Explorer view, right-click the **FurnaceStatus** project and then click **Deploy**.

 **Start this tutorial:** “Task 1: Creating an event project.”

Related information:

“Tutorial scenario” on page 64

You can track the events related to a central heating system in a smart home by using Decision Server Events. In this tutorial scenario, the owner of a smart home uses events to record and track the state of the central heating furnace, such as starting, running, shutting down, or stopped.

Task 1: Creating an event project

In this task, you create an event project to contain the tutorial application. In the event project, you add four furnace tracking events, one for each of the furnace operating states. The project also includes an event object for each event, along with several actions and corresponding action objects.

When you complete this task, these event and action objects are added to the event project:

- The **Events** category contains these furnace events:
 - Furnace is starting
 - Furnace is running
 - Furnace is stopping
 - Furnace is stopped

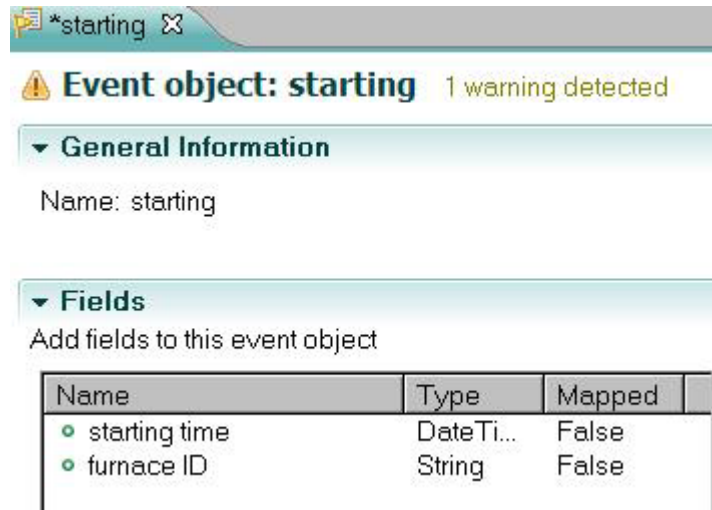
Each of the events includes an event object that represents the operating state of the furnace such as starting, running, stopping, and stopped.

- The **action** category contains these actions that are triggered by the events:
 - Alert furnace is running notice without furnace is starting
 - Alert furnace is stopped notice without furnace is stopping
 - Alert incomplete furnace start
 - Alert incomplete furnace stop

Each action includes an action object with fields for the furnace operating times, such as start time, stop time and running time.

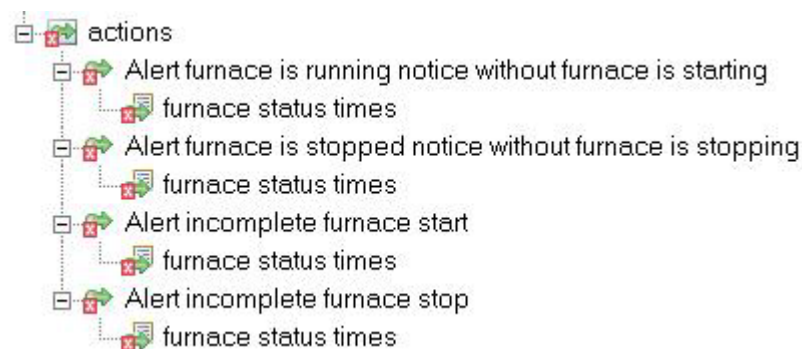
1. Create an event project to contain the tutorial application.
 - a. Right-click in the Event Explorer view, then click **New > Event Project**.
 - b. Enter FurnaceStatus as the name of the project.
 - c. Click **Finish**.
2. Create four furnace events.
 - a. Right-click the **assets** folder, then click **New > Event**.
 - b. In the New Event wizard, select **Create a blank event** and click **Next**.
 - c. Enter Furnace is starting as the name of the first event, then click **Finish**. The wizard closes and the event is opened in the Event Editor.
 - d. Repeat the steps to create three additional events:
 - Furnace is running
 - Furnace is stopping
 - Furnace is stopped
3. Create an event object for each furnace event.
 - a. In the Event Editor, open the **Furnace is starting** event.
 - b. In the Event Objects section, click **Add**. The New Event Object wizard is displayed.
 - c. Select **Add a new blank event object**, then click **Next**.

- d. Enter **starting** as the name of the event object, then click **Finish**.
 - e. Repeat the steps for each event to create the following event objects that correspond with the furnace events:
 - running event object for the Furnace is running event
 - stopping event object for the Furnace is stopping event
 - stopped event object for the Furnace is stopped event
 - f. Click **File > Save All** to save the changes.
4. Add a time field and furnace ID field to each of the event objects.
- a. In the Furnace is starting Event Editor window, double-click the starting event object to open it in the Event Object Editor window.
 - b. In the Event Object Editor, in the Fields section, click **Add**.
 - c. In the New Field wizard, name the first field starting time, then select **DateTime** as the data type.
 - d. Click **Finish**.
 - e. Repeat the steps to add a furnace ID field with the data type **String**.
 - f. Click **File > Save**. This example shows the **starting** event object with the added fields:




- g. Repeat the steps for each event, opening the associated event object and adding the following fields and data types.
 - Add the stopping time field to the stopping event object, with the **DateTime** data type.
 - Add the furnace ID field to the stopping event object, with the **String** data type.
 - Add the running time field to the running event object, with the **DateTime** data type.
 - Add the furnace ID field to the running event object, with the **String** data type.
 - Add the stopped time field to the stopped event object, with the **DateTime** data type.
 - Add the furnace ID field to the stopped event, with the **String** data type.
- h. Click **File > Save All**.
- i. Click **File > Close All** to close the Event Object Editor windows and the Event Editor windows.


5. Create the actions that are triggered if the events are not received in the expected order, or at the specified time, for example, if the furnace running event is received before the furnace starting event.
 - a. Right-click the **assets** folder, then click **New > Action**.
 - b. In the New Action wizard, select **Create a blank action** and click **Next**.
 - c. Enter Alert furnace is running notice without furnace is starting as the name of the first action, then click **Finish**. The wizard closes and the action is opened in the Action Editor.
 - d. Repeat the steps to create these actions:
 - Alert furnace is stopped notice without furnace is stopping
 - Alert incomplete furnace start
 - Alert incomplete furnace stop
6. Add an action object to each of the furnace events.
 - a. Click the Action Editor window to edit the Alert furnace is running notice without furnace is starting action.
 - b. In the Action Objects section, click **Add**. The New Action Object wizard is displayed.
 - c. Select **Add a new blank action object**, then click **Next**.
 - d. Enter **furnace status times** as the name of the first action object, then click **Finish**.
 - e. Click the Action Editor window to edit the Alert furnace is stopped notice without furnace is stopping action.
 - f. In the Action Objects section, click **Add**.
 - g. Select the **Share an existing action object** option, then select **furnace status times** from the **Select action object** list.
 - h. Repeat the steps to share the furnace status times action object with the two remaining actions, Alert incomplete furnace start and Alert incomplete furnace stop.
 - i. Click **File > Save All**. The following example shows the **FurnaceStatus** project in the Event Explorer view:



7. Add five fields to the shared furnace status action object: start time, running time, stop time, stopped time, and furnace ID.
 - a. In the Action Editor, in the Action Objects section, click **furnace status times**, then click **Edit**.
 - b. In the Action Object Editor, in the Fields section, click **Add**.
 - c. In the New field wizard, enter start time as the name of the first field, then select **DateTime** as the data type.
 - d. Click **Finish**.

- e. Repeat the steps to add these fields and data types:
 - running time, data type **DateTime**
 - stop time, data type **DateTime**
 - stopped time, data type **DateTime**
 - furnace ID, data type **String**
- f. Click **File > Save**. This example shows the action object containing data fields for the furnace status times, and a furnace ID field for the context:

furnace status times 






 **Action object: furnace status times** 1 error detected


General Information

Name: furnace status times

Fields

The fields in the action are grouped into action objects, which can be of actions.

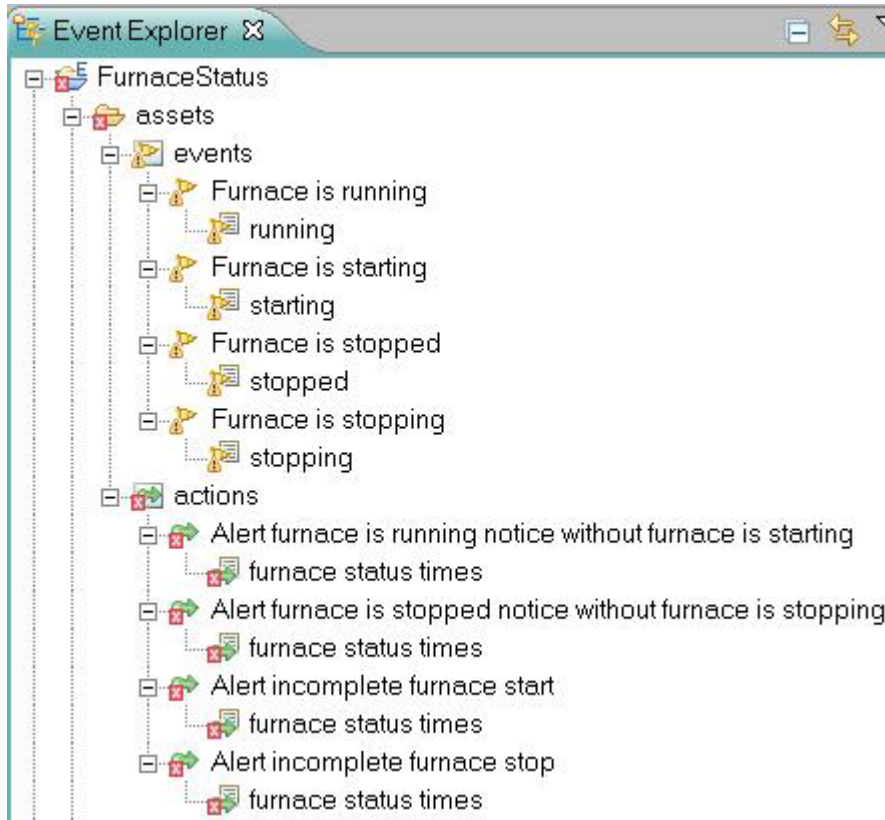
Name	Type	Definition Ty...	Definition
 furnace ID	String	Undefined	
 start time	DateTi...	Undefined	
 running time	DateTi...	Undefined	
 stop time	DateTi...	Undefined	
 stopped time	DateTi...	Undefined	

 At least one field must have a definition.

Note: You can ignore the error message in the Action Object Editor window because it is resolved during a later stage in the tutorial.

- g. Close the Action Object Editor and the Action Editor.

In the Event Explorer view, the **FurnaceStatus** project contains events and actions, and objects that work with them, as shown in this example:



In the next task, you create the business objects that contains the tracking data and the context ID.

 **Next:** “Task 2: Defining the business objects”

Task 2: Defining the business objects

In this task, you create a business object to hold the activity data which records the state of the furnace, and a second business object to hold the context ID. The first business object must be configured as a summary instance object.

1. Create a new business object named Furnace status. Configure this business object as a summary instance.
 - a. Right-click the **assets** folder, then click **New > Business Object**.
 - b. Enter Furnace status as the name of the business object, then click **Next**.
 - c. In the New Business Object window, select **Start with a blank business object**, then click **Finish**. The new business object opens in the Business Object Editor.
 - d. In the Context-Scoped Settings section, select the option **Maintains a single set of values based on a calculation of values from multiple events**.
 - e. In the same section, select the option **Keep the values in this object persistent across system shutdowns**. This example shows the Context-Scoped Settings section with the correct options selected:

▼ Context-Scoped Settings

Configure this business object to store values received across multiple events within a context.

- Contains data received from a single event.
 - Specify how many instances of the business object exist at run time in response to the incoming event.
 - Always treat as a single instance (scalar)
 - Always treat as multiple instances (array)
 - Can be either
- Maintains a single set of values based on a calculation of values from multiple events (summary instance).
- Maintains an array of values from multiple events (accumulating array).
 - Specify the number of events over which to retain values in the array.

Maximum number of occurrences
 - Specify the period of time over which to retain values in the array.

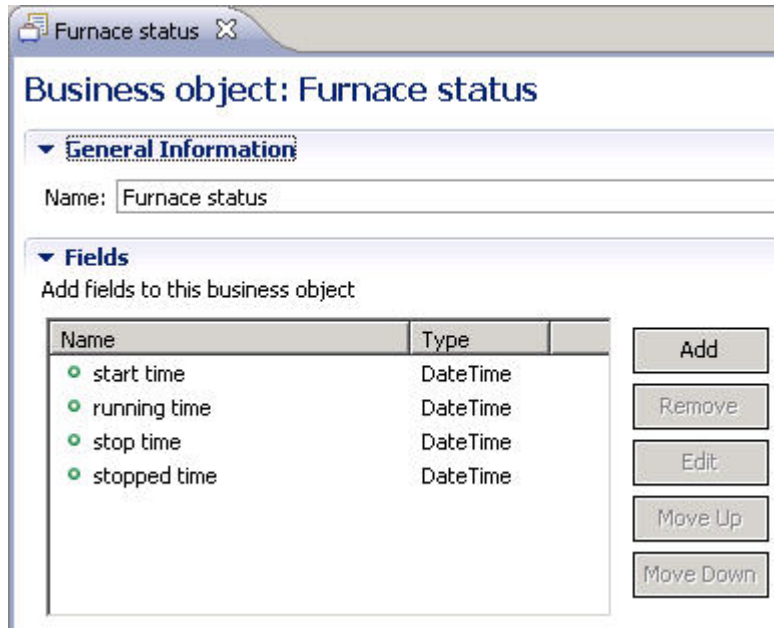
Days:

Hours:

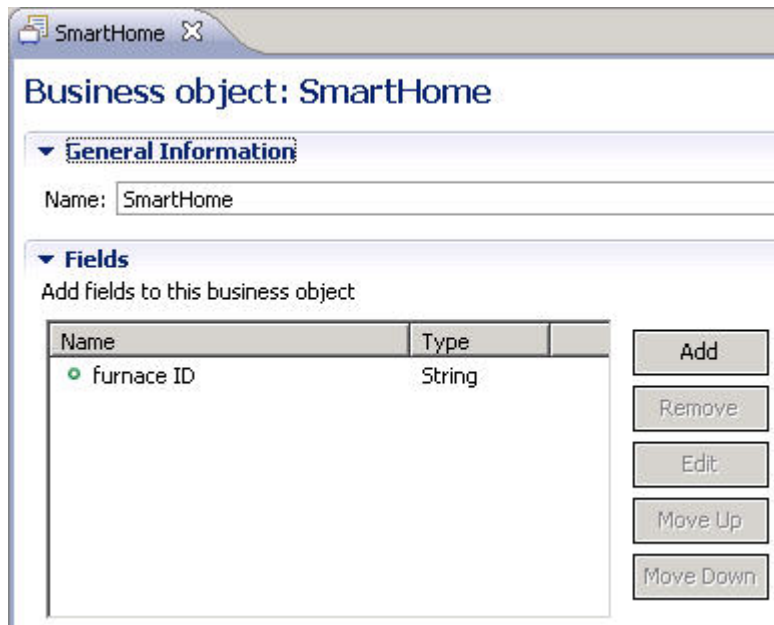
Minutes:

Seconds:
- Keep the values in this object persistent across system shutdowns.

- f. Click **File > Save**.
2. Define four business object fields in the Furnace status business object to contain the furnace operating states.
 - a. In the Business Object Editor, in the Fields section, click **Add**.
 - b. In the New field window, enter start time as the new field name, and select **DateTime** as the data type.
 - c. Click **Finish**.
 - d. Repeat the steps to add these fields and data types:
 - running time, data type **DateTime**
 - stop time, data type **DateTime**
 - stopped time, data type **DateTime**
 - e. Click **File > Save**. This example shows the Furnace status business object with the correct fields added:



3. Create a second business object named SmartHome to hold the context information, which includes the furnace ID.
 - a. Create a new business object called SmartHome.
 - b. Using the Business Object Editor, define a business object field called **furnace ID** with a data type of **String**. This example shows the SmartHome business object with the correct field added:



4. Verbalizations are created automatically for business objects and business object fields. Edit the default verbalizations to improve the phrasing for these assets that you use later in the tutorial to create filters and events.
 - a. In the Business Object Editor, open the Furnace status object.
 - b. In the Fields section, double-click **start time** to edit the field. The start time field properties are displayed in the Field window.

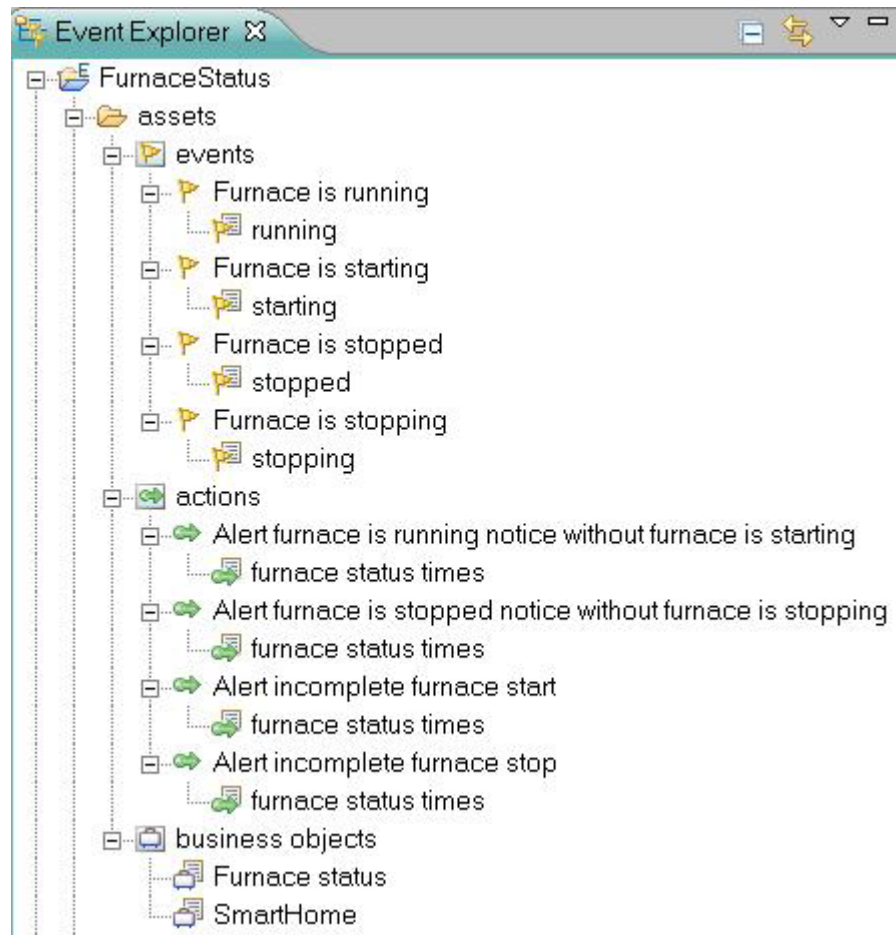
- c. In the Verbalization section of the window, verify that the default verbalization {start time} of {this}, has been added to the field properties.
 - d. Click the **Overview** tab to return to the business object Fields table.
 - e. Repeat the steps to verify the default verbalizations for the remaining fields in the Furnace status business object.
 - f. Click the **Overview** tab to return to the Furnace status business object window, and verify that the default verbalization of the business object is "furnace status" This verbalization creates awkward English phrasing. The phrasing is clearer if it reads simply "furnace" In the next step, you change the template used to build the verbalization, which improves the phrasing.
 - g. In the **Verbalization** section, change the term from "furnace status" to "furnace"
 - h. Click **File > Save**.
 - i. Close the Business Object Editor window.
5. Add field constructors to define how the business object fields in the two business objects are populated. Field constructors define the mapping from the event object to the business object. The mapping moves data from a field in the event object to a field in the business object so that each field receives its value from the corresponding event object field.
 - a. In the Event Explorer view, open the **assets > Events** folder and double-click **Furnace is running** to open the event in the Event Editor.
 - b. In the Event Objects section, double-click **running** to open the event object in the Event Object Editor.
 - c. In the Field Constructors section, click **Add**.
 - d. In the Field Constructor window, expand the business objects to display all of the business object fields. Select **running time** and **furnace ID**, then click **Finish**.
 - e. In the field constructors table, double-click the **Furnace status > running time** row to map this business object field to the event object.
 - f. In the Field Constructor window, in the Definitions section, click **Definition type**, then select **Field**.
 - g. Select **running time** as the Field name.
 - h. Click the **Overview** tab to return to the Field constructors section.
 - i. Double-click the **SmartHome > furnace ID** row in the field constructors table.
 - j. In the Field Constructor window, click **Definition Type** in the Definitions section, then select **Field**.
 - k. Select **furnace ID** as the Field name.
 - l. Repeat these steps for each event to add the relevant field constructor for each event object field, that matches the furnace status in the event:
 - map the **starting time** field to the **starting** event object
 - map the **stopped time** field to the **stopped** event object
 - map the **stop time** field to the **stopping** event object
 - m. In addition, map the **furnace ID** field to each event object. The following example shows the **starting** event object with the correct field constructors added:

Field Constructors				
Add business objects that will receive data from the fields in this event object.				
Business Object	Field	Data Type	Definition Type	Definition Value
Furnace status	start time	DateTime	Field	starting time
SmartHome	furnace ID	String	Field	furnace ID


- n. Click **File > Save All**.
 - o. Close the Event Object Editor windows.
6. Add business object definitions to the action object fields. Action object fields must receive values from the business objects so that the required tracking data is added to the action. If an action object field does not include a field definition, then the value of the field is null.
- a. In the **assets** folder, expand **Actions > Alert incomplete furnace start**, then double-click **furnace status times** to open the action object in the Action Object Editor.
 - b. In the Fields section, double-click **furnace ID** to edit the field.
 - c. In the Definition section, select **Field** from the Definition Type list.
 - d. Select **SmartHome** as the business object and **furnace ID** as the business object field. As a result of the defined relationship between the business object and the action object, the value from the selected business object is passed directly to the action field when the tutorial application is deployed.
 - e. Click the **Overview** tab to return to the furnace status times Fields section.
 - f. In the Fields table, double-click **start time** to edit the field.
 - g. Select **Field** as the Definition Type.
 - h. Select **Furnace status** as the business object and **start time** as the business object field.
 - i. Repeat the steps for each of the fields to add a business object definition to the action object field. The following example shows the furnace status action object with the correct business object definition added to the stopped time field:

- j. Click **File > Save**.
- k. Close the Action Object Editor.

In the Event Explorer view, the **FurnaceStatus** event project is shown in the following example:



In the next task, you define the event rule that evaluates the furnace status tracking data.

 **Next:** "Task 3: Defining the event rules and filter"

Related concepts:

Field constructors

Related tasks:

Defining business objects

Defining business object fields

Task 3: Defining the event rules and filter

Using Event Designer, you can define event rules that track the operational state of the smart home furnace. You also create filters to determine whether the event for a specific operational state is received within a specified time.

In this tutorial task, you define the event rules that track the furnace operating state, and the filters that determine if a furnace event happened within a specific elapsed time.

1. Create the filter to check whether an event related to the furnace operating state is received within a specific time period.

- a. Right-click the **assets** folder, then click **New > Filter**.
- b. Enter **Furnace is not running** as the name of the first filter, then click **Finish**. The wizard closes and the filter is opened in the Filter Editor.
- c. In the Filter Editor window, click to place the edit cursor inside the **Content** field.
- d. Press **Ctrl+Spacebar** to display the Content Assist box.
- e. Scroll down in the Content Assist box until you see the phrase **past occurrences of <an event or action>** .
- f. Double-click to select **past occurrences of <an event or action>** from the list and add this phrase to the filter content field.
- g. Select **furnace Is running** from the list.
- h. Using the Content Assist box, add the phrase **within <duration>**.
- i. Click the **<duration>** variable and replace the variable with the value 30 seconds.
- j. Place the cursor after the 30 seconds value and press **Ctrl+Spacebar**, then select **equals <a number>** from the list.
- k. Select the **<a number>** variable, then enter zero (0) as the final filter value. The following example shows the completed filter phrase:
past occurrences of furnace Is running within 30 seconds equals 0
- l. Repeat the steps to create these filters:

Furnace has not stopped

past occurrences of furnace Is stopped within 30 seconds
equals 0

No furnace start was issued

past occurrences of furnace Is starting within 30 seconds
equals 0

No furnace stop was issued

past occurrences of furnace Is stopping within 30 seconds
equals 0

- m. Click **File > Save All**, then close the Filter Editor windows.
2. Create the event rules that use the filters.
 - a. Right-click the **assets** folder, then click **New > Event Rule**.
 - b. In the New Event Rule wizard, enter **Check furnace state after running notification** as the name of the event rule.
 - c. Click **Next**.
 - d. In the Select Event panel, click **furnace Is running** to indicate that the new event rule applies to the selected event.
 - e. Click **Next**.
 - f. In the Define Context Relationship panel, select **the furnace ID of the smartHome** as the context so that the event rule can use the furnace ID to reference the summary business object.
 - g. Click **Finish**. The event rule is opened in the Rule Editor.
 - h. To add the event rule content, click inside the **Content** field to place the edit cursor in the field.
 - i. Press **Ctrl+Spacebar** to open the Content Assist box.
 - j. Select **if <conditions>** from the list.
 - k. Click **no furnace start was issued**.
 - l. Click **then <actions>**.

- m. Click **alert furnace Is running notice without furnace Is starting**.
- n. Click the semicolon symbol (;), which indicates the end of the rule content.
The following example shows the completed rule event:

Content
Directly type in this section to create or modify your event rule definition

```

if no furnace start was issued
then alert furnace Is running notice without furnace Is starting ;

```

- o. Repeat the steps to create the following event rules.

New event rule	Associated event	Rule content
Check furnace state after starting notification	furnace Is starting	after 30 seconds if furnace Is Not running then alert incomplete furnace start ;
Check furnace state after stopped notification	furnace Is stopped	if no furnace stop was issued then alert furnace Is stopped notice without furnace is stopping ;
Check furnace state after stopping notification	furnace Is stopping	after 30 seconds if furnace has Not stopped then alert incomplete furnace stop ;

- p. Save the event rules and close the Event Editor windows.

In the next task, you deploy the tutorial application.

 **Next:** “Task 4: Deploying the application”

Task 4: Deploying the application

In this task, you deploy the tracking tutorial application.

Now that you have completed the event project that defines the tutorial application, you must deploy the event project to the event run time that is running on your system so that you can confirm that the application works.

1. In the Event Explorer window, right-click the **FurnaceStatus** project name, then click **Deploy**.
2. In the Deploy window, check to make sure that the default option **Deploy all assets** is selected, then click **Next**.
3. Select an existing event run time from the Deploy window, then click **Finish**.
The event project assets are deployed to the event run time.

If the deployment fails, an error message is displayed in the Problems view. Check that you have defined the assets in the event project as described in the previous tutorial tasks.

You have completed the tutorial. Use the Event Tester widget to test the behavior of the sample event project by submitting events.

Related tasks:

Testing event logic using the Event Tester widget

Tutorial: Maintaining a running total

This tutorial shows you how to maintain a running total that accumulates a value from events using Decision Server Events.

You can track the accumulated value of a series of events to maintain a running total. The value total is tracked and when the total reaches a specified limit, an action is triggered.

Before you start

In this tutorial, you create an event project to track the running total of extra minutes that a customer purchases, beyond the prepaid mobile phone plan minutes, in a month.

To maintain a running total of mobile phone minutes using Decision Server Events, you must define a summary instance object. The fields in the business object are used to track total phone minutes for the month. You define a filter and an event rule to specify the running total limit that triggers an alert action. The action sends a message to the customer that phone usage has exceeded the prepaid minutes for the month, and suggesting that the customer might want to upgrade the mobile phone plan.

In this tutorial you complete the following tasks:

- “Task 1: Creating an event project” on page 80
- “Task 2: Defining the business objects” on page 83
- “Task 3: Defining the event rule and filter” on page 88
- “Task 4: Deploying the application” on page 90

What you will learn

In this tutorial, you will learn these skills:

- Use Decision Server Events to maintain a running total that accumulates a value from events.
- Configure a business object as a summary instance object to collect and store data about the mobile phone transactions.
- Define field constructors to populate the summary object fields.
- Create an action that is triggered if the running total of extra mobile phone minutes exceeds a specified amount.
- Create a filter to specify the running total limit.

This tutorial takes about 2 hours to complete.

What you need to know

This tutorial is intended for developers who want to learn how to develop event projects in Decision Server Events. The procedures in the tutorial assume that you are familiar with the Eclipse environment, and that you have completed the Getting started with event rules tutorial.

 **Start this tutorial:** “Task 1: Creating an event project” on page 80.

Tutorial scenario

You can track the accumulated value of a series of events to maintain a running total. The value total is tracked and when the total reaches a specified limit, an action is triggered. In this tutorial scenario, a telecommunications company tracks how much a customer spends on extra minutes beyond the minutes provided in the mobile phone plan. If the total of extra minutes exceeds a specified threshold, the company sends the customer an offer to upgrade the plan.

In various business scenarios, a running total is required to accumulate a value from events. The total value can then be tracked and when a specific limit is reached the total triggers an action. Businesses can use this total to customize offers to fit the individual needs of the customers, to reward customers for their loyalty, or to warn of any potential security risks concerning the spending habits of the customer. Using Decision Server Events, the telecommunications company can track how much money a customer is spending on extra minutes for a mobile phone, and offer the customer a different plan that might include more free minutes to help the customer save money.

To complete the tutorial, you complete these tasks:

1. Create an event project and add an event to maintain the running total of mobile phone minutes throughout the month, and at the start of each month, reset the total to zero.
2. Create a summary instance business object to hold the summary data that includes the running total, and another business object that contains customer data.
3. Create a filter to specify the running total limit.
4. Define the event rule that compares the running total with the specified limit, and triggers an action if the limit is exceeded.
5. Deploy the application created by the completed tutorial.

 **Start this tutorial:** “Task 1: Creating an event project” on page 80.

Related information:

“Running the completed tutorial”

You can import, view and run the tutorial in its completed state. When completed, the tutorial contains an event project that you can run to maintain a running total of the extra mobile phone minutes used by a telecommunications customer.

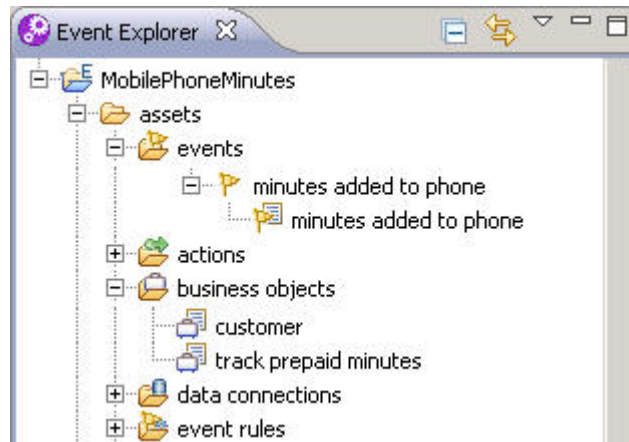
Running the completed tutorial

You can import, view and run the tutorial in its completed state. When completed, the tutorial contains an event project that you can run to maintain a running total of the extra mobile phone minutes used by a telecommunications customer.

To import the completed project and run the tutorial application:

1. Download the completed tutorial project for tracking the system state. Right-click `MobilePhoneMinutes.xml` and save the file to your local system.
2. In the Event Explorer view, right-click and select **Import**.
3. In the Import wizard window, click **Event Designer > Event Project from XML File**, then click **Next**.
4. Click **Browse** to select the event project file to import: `MobilePhoneMinutes.xml`.


5. Navigate to the location of the MobilePhoneMinutes.xml file, select the file, click **Open**, then click **Next**.
6. Click **Next** in the Project Assets panel.
7. If you already have a project that uses the default project name, **MobilePhoneMinutes**, change the project name in the **New event project name** field.
8. Click **Finish**. The imported project appears in the Event Explorer view:



The project contains an event that records the extra minutes added to the mobile phone, and an associated event object with the same name. Two business objects, track prepaid minutes and customer, contain the total minutes summary data and the unique customer ID. In the **Actions** folder, the project contains an alert action that is triggered if the specified number of extra minutes is exceeded.

To run the completed tutorial:

1. Make sure that you are in the Event perspective. Click **Window > Open Perspective > Other > Event**.
2. In the Event Explorer, double-click event project items to view them in the default editors.
3. Deploy the MobilePhoneMinutes project to the event run time. In the Event Explorer view, right-click the **MobilePhoneMinutes** project, then click **Deploy**.

 **Start this tutorial:** “Task 1: Creating an event project.”

Related information:

“Tutorial scenario” on page 79

You can track the accumulated value of a series of events to maintain a running total. The value total is tracked and when the total reaches a specified limit, an action is triggered. In this tutorial scenario, a telecommunications company tracks how much a customer spends on extra minutes beyond the minutes provided in the mobile phone plan. If the total of extra minutes exceeds a specified threshold, the company sends the customer an offer to upgrade the plan.

Task 1: Creating an event project

In this task, you create an event project to contain the running total tutorial application. In the event project, you add a phone minutes event that contains the mobile phone transaction data such as minutes added and activity date.

Create the phone minutes event, the associated event object, and event fields in the object. Define the event fields to contain details of the phone minutes transaction and the context ID.

When you complete this task, an event and action are added to the event project:

- The **Events** category contains the minutes added to phone event.
- The **action** category contains the alert customer suggest new plan action that is triggered by the event.

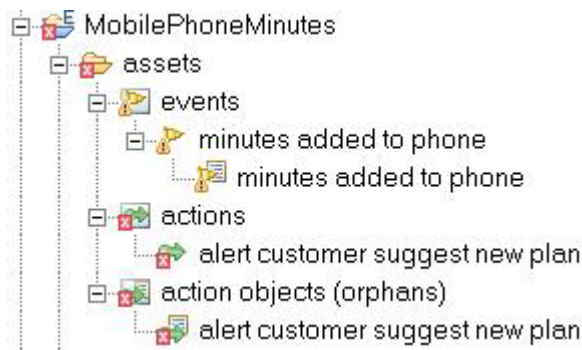
1. Create an event project to contain the tutorial application.
 - a. Right-click in the Event Explorer view, then click **New > Event Project**.
 - b. Enter **MobilePhoneMinutes** as the name of the project.
 - c. Click **Finish**.
2. Create the minutes added to phone event.
 - a. Right-click the **assets** folder, then click **New > Event**.
 - b. In the New event wizard, select **Create a blank event**, then click **Next**.
 - c. Enter minutes added to phone as the name of the first event, then click **Finish**. The wizard closes and the event is opened in the Event Editor.
3. Create a minutes added to phone event object for the phone event.
 - a. In the Event Objects section of the Event Editor window, click **Add**. The New Event Object wizard is displayed.
 - b. Select **Add a new blank event object**, then click **Next**.
 - c. Enter minutes added to phone as the name of the event object, then click **Finish**.
4. Add event object fields to the event object.
 - a. In the Event Object Editor, in the Fields section, click **Add**.
 - b. In the New field wizard, enter mobile number as the name of the first field, then select **String** as the data type.
 - c. Click **Finish**.
 - d. Repeat the steps to add these fields:
 - **minutes added** with the **Integer** data type
 - **activity date** with the **DateTime** data type

The following example shows the minutes added to phone event object with the added event object fields:



- e. Click **File > Save All**.
- f. Close the Event Object Editor window.

5. Create the action that is triggered if the total of purchased mobile phone minutes exceeds the specified limit.
 - a. Right-click the **assets** folder, then click **New > Action**.
 - b. In the New Action wizard, select **Create a blank action**, then click **Next**.
 - c. Enter alert customer suggest new plan as the name of the first action, then click **Finish**. The wizard closes and the action is opened in the Action Editor.
6. Add an action object to the action.
 - a. In the Action Objects section of the Action Editor window, click **Add**. The New Action Object wizard is displayed.
 - b. Select **Add a new blank action object**, then click **Next**.
 - c. Enter alert customer suggest new plan as the name of the action object, then click **Finish**. This example shows the action category with the correct action and an associated action object:



Note: You can ignore the error message in the Action Object Editor window because it is resolved during a later stage in the tutorial.

7. Add three action object fields to the action object.
 - a. In the Action Objects section of the Action Editor, click **alert customer suggest new plan**, then click **Edit**.
 - b. In the Fields section, click **Add**.
 - c. In the New field wizard, name the first field mobile number, then select **String** as the data type.
 - d. Click **Finish**.
 - e. Repeat the steps to add these fields and data types:
 - activity date, data type **DateTime**
 - total minutes for month, data type **Integer**

This example shows the action object with three data fields to hold the phone minutes transaction data:

*alert customer suggest new plan ✕

✕ **Action object: alert customer suggest new plan** 1 error detected

▼ **General Information**

Name: alert customer suggest new plan

▼ **Fields**

The fields in the action are grouped into action objects, which can be optionally reused by other actions.

Name	Type	Definition Ty...	Definition Val...
mobile number	String	Undefined	
activity date	DateTi...	Undefined	
total minutes for month	Integer	Undefined	

Add...
Remove
Edit
Move Up
Move Down

At least one field must have a definition.

Note: The Definition Type and Definition Value for each field are currently undefined. The next tutorial task includes the procedure to create the definitions.

- f. Click **File > Save**.
- g. Close the Action Object Editor and the Action Editor.

In the Event Explorer view, the **MobilePhoneMinutes** project contains an event and an action, and objects that correspond to the event and action, as shown in this example:



In the next task, you create the business objects which contain the phone minutes transaction data and the context ID.

▶ **Next:** “Task 2: Defining the business objects”

Task 2: Defining the business objects

Create a business object to hold the phone minutes transactions summary data for the running total, and a second business object to hold the customer context ID. The transactions business object must be configured as a summary instance object.

1. Create a new business object named track prepaid minutes to track the total number of extra minutes purchased for the mobile phone during the month. Configure this business object as a summary instance.
 - a. Right-click the **assets** folder, then click **New > Business Object**.
 - b. Enter track prepaid minutes as the name of the business object, then click **Next**.
 - c. In the New Business Object window, select **Start with a blank business object**, then click **Finish**. The new business object opens in the Business Object Editor.
 - d. In the Context-Scoped Settings section, select the option **Maintains a single set of values based on a calculation of values from multiple events**. The following example shows the Context-Scoped Settings section with the correct option selected:

▼ Context-Scoped Settings

Configure this business object to store values received across multiple events within a context.

Contains data received from a single event.
Specify how many instances of the business object exist at run time in response to the incoming event.

- Always treat as a single instance (scalar)
- Always treat as multiple instances (array)
- Can be either

Maintains a single set of values based on a calculation of values from multiple events (summary instance).

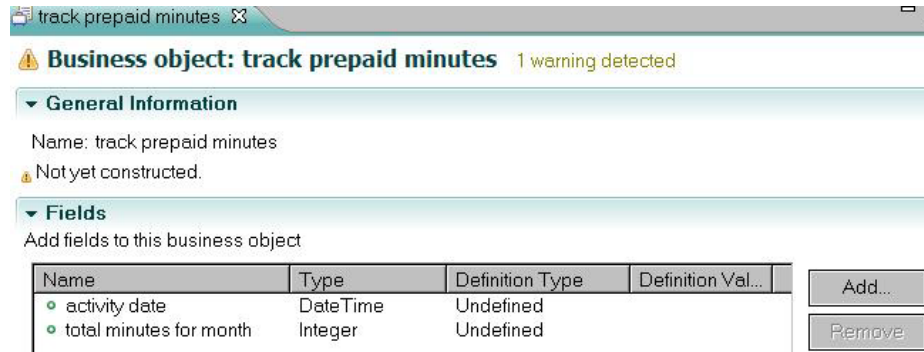
Maintains an array of values from multiple events (accumulating array).

- Specify the number of events over which to retain values in the array.
Maximum number of occurrences:
- Specify the period of time over which to retain values in the array.
Days:
Hours:
Minutes:
Seconds:

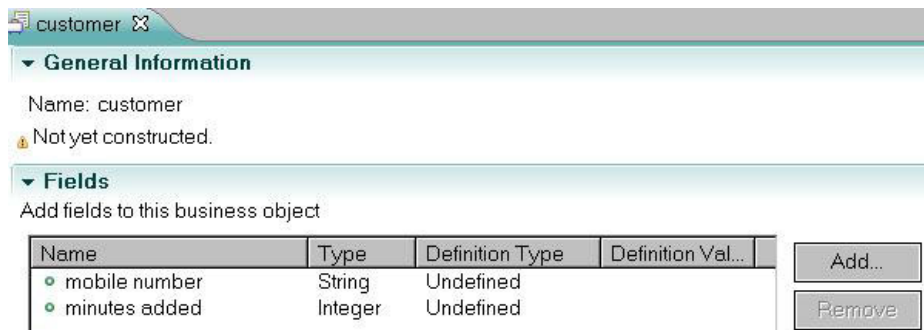
Keep the values in this object persistent across system shutdowns.

- e. Click **File > Save**.
2. Define two business object fields in the business object to contain the activity data and the running total of purchased mobile phone minutes.
 - a. In the Fields section, click **Add**.

- b. In the New field window, enter activity date as the name of the first field, then select **DateTime** as the data type.
- c. Click **Finish**.
- d. Repeat the steps to add the total minutes for month field with the **Integer** data type. The following example shows the business object with the correct fields added:



- e. Click **File > Save**.
3. Create the customer business object to hold the context information, which includes the customer ID.
 - a. Create a new business object called customer.
 - b. Define a business object field called mobile number with the **String** data type.
 - c. Add a second business object field called minutes added with the data type **Integer**. The following example shows the business object with the correct fields added:



4. Verbalizations are created automatically for business objects and business object fields.
 - a. In the Business Object Editor, open the customer business object.
 - b. In the Fields section, double-click **mobile number** to edit the field. The mobile number field properties are displayed in the Field window.
 - c. In the Verbalization section of the window, verify that the default verbalization {mobile number} of {this}, has been added to the field properties.
 - d. Click the **Overview** tab to return to the business object Fields table.
 - e. Repeat the steps to verify the default verbalization for the **minutes added** field.
 - f. Click the **Overview** tab to return to the customer business object and verify that the default verbalization of the business object is customer.
 - g. Click **File > Save All**.

- h. Close the Business Object Editor windows.
5. Add field constructors to define how the business object fields in the two business objects are populated. Field constructors define the mapping from the event object to the business object. The mapping moves data from a field in the event object to a field in the business object so that each field receives its value from the corresponding event object field.
 - a. In the **assets** folder, open the **Events** category, then double-click the **minutes added to phone** event to open it in the Event Editor.
 - b. In the Event Objects section, double-click the **minutes added to phone** event object to open it in the Event Object Editor.
 - c. In the Field Constructors section, click **Add**.
 - d. Click **Select all**, then click **Finish**.
 - e. In the Field constructors table, double-click the **customer > mobile number** row to map this business object field to the event object.
 - f. In the Definitions section, click **Definition type**, then select **Field**.
 - g. In the Event Object field, select **mobile number**. The mobile number business object field is populated by mapping to the corresponding field in the event object.
 - h. Click the **Overview** tab to return to the Field constructors section.
 - i. Double-click the **customer > minutes added** row.
 - j. In the Definitions section, click **Definition type**, then select **Field**.
 - k. In the Event Object field, select **minutes added**.
 - l. Repeat the steps to add a field constructor for the **track prepaid minutes > activity date** business object field that matches the activity date in the event object.
 - m. Click the **Overview** tab to return to the Field constructors section.
 - n. Double-click the **track prepaid minutes > total minutes for month** row to edit the field constructor.
 - o. In the Definitions section, select **JavaScript**, then click in the **JavaScript expression** field. The **total minutes for the month** business object field contains the running total of extra minutes that the customer purchased for the mobile phone. To calculate the correct value for this field, the field constructor uses JavaScript to clear the previous value, if required, and then add minutes from the event to the summary business object. The following example shows the JavaScript that you add to the field constructor to calculate the total minutes for the current month.


```

if (track prepaid minutes.total minutes for month == null) {
tempMinutes = 0;
}
else {
tempMinutes = track prepaid minutes.total minutes for month;
}

if (track prepaid minutes.activity date == null) {
tempMinutes = 0;
}
else if (track prepaid minutes.activity date.getMonth() != activity date.getMonth()) {
tempMinutes = 0;
}
tempMinutes + minutes added;

```

This JavaScript performs the following functions:

- Checks to determine whether this event is the first occurrence of tracking minutes for this mobile phone. If it is, then the value of the variable

track prepaid minutes.total minutes for month is null. If the variable is null, then set the tempMinutes variable to zero.

- If track prepaid minutes.total minutes for month is not null, then sets the tempMinutes variable to the current value of the running total for the month.
- Checks to determine if this activity date represents the first time that minutes are tracked for this mobile phone. If it is, then the variable track prepaid minutes.activity date is null. A null value for a date is not valid, therefore, initializes the variable tempMinutes by setting it to zero.
- Checks to determine if the customer has added extra minutes to the phone in the same month as the previous phone minutes transaction.
- If the customer has not added extra minutes in the same month, then sets the variable tempMinutes back to zero. If the customer has purchased extra minutes, then adds the minutes to the running total.

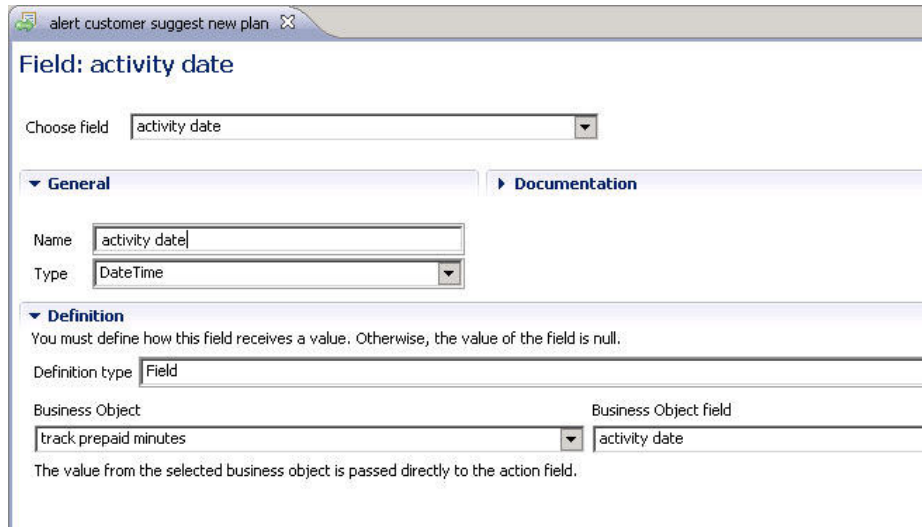
When creating a summary object, you can generate basic JavaScript in example field constructors.

- p. Click **File > Save All**. This example shows the **minutes added to phone** event object with the correct field constructors added:

Field Constructors				
Add business objects that will receive data from the fields in this event object				
Business Object	Field	Data Type	Definition Type	Definition Value
customer	mobile number	String	Field	mobile number
customer	minutes added	Integer	Field	minutes added
track prepaid minutes	activity date	DateTime	Field	activity date
track prepaid minutes	total minutes for month	Integer	Javascript	if (track prepaid minutes.tot...

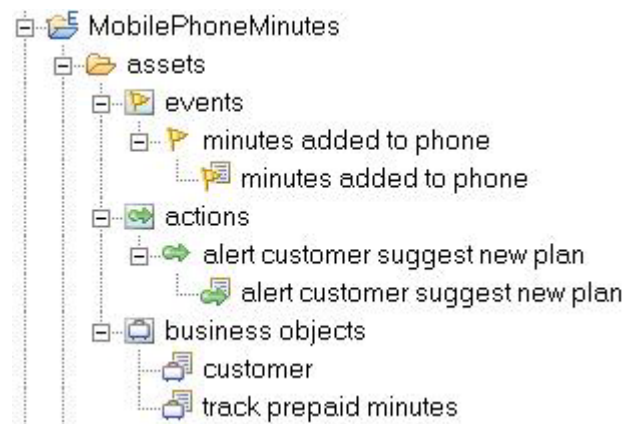
If the date in the activity date field in the minutes added to phone event is the same month as the activity date in the summary object, the value of the minutes added field from the event is added to the total minutes for month business object field for the current month. If the date in the activity date field in the minutes added to phone event is not the same month as the activity date in the summary object, the total minutes for month field is set to the value of the minutes added field from the inbound event.

- q. Close the Event Object Editor windows.
6. Add business object definitions to the action object fields. Action object fields must receive values from the business objects so that the required running total data is added to the action. If an action object field does not include a field definition, then the value of the field is null.
- In the **assets** folder, expand **Actions > alert customer suggest new plan**, then double-click **alert customer suggest new plan** to open the action object in the Action Object Editor.
 - In the Fields section, double-click **mobile number**.
 - In the Definition section, in the Definition Type field, select **Field**.
 - Select **customer** as the Business Object and **mobile number** as the Business Object Field. As a result of the defined relationship between the business object and the action object, the value from the selected business object is passed directly to the action field when the tutorial application is deployed.
 - Click the **Overview** tab to return to the Fields section.
 - In the Fields section, double-click **activity date**.
 - In the Definition section, in the Definition Type field, select **Field**.
 - Select **track prepaid minutes** as the Business Object and **activity date** as the Business Object Field. The following example shows the alert customer suggest new plan action object with the correct business object definition added to the activity date business object field:



- i. Repeat the steps to add a definition for the total minutes for month field, selecting **track prepaid minutes** as the Business Object and **total minutes for month** as the Business Object Field.
- j. Click **File > Save**.
- k. Close the Action Object Editor window.

In the Event Explorer view, the following example shows the **MobilePhoneMinutes** event project:



In the next task, you define the event rule that compares the running total with the specified alert trigger limit.

Next: “Task 3: Defining the event rule and filter”

Related concepts:

Field constructors

Sharing data across events by creating a summary object

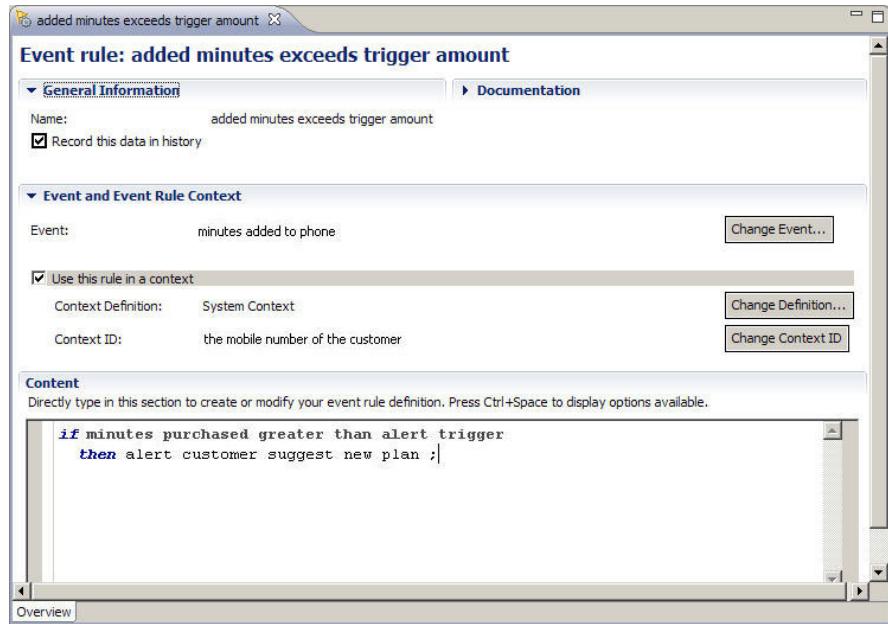
Task 3: Defining the event rule and filter

In this tutorial task, you define an event rule that checks the running total of extra mobile phone minutes purchased by a telecommunications customer. The rule

compares the running total with the specified limit and triggers an action to contact the customer if the limit is exceeded. You also create a filter that specifies the running total limit.

1. Create the filter to determine the extra minutes limit. When the limit is exceeded, the rule event triggers the alert action.
 - a. Right-click the **assets** folder, then click **New > Filter**.
 - b. Enter **minutes purchased greater than alert trigger** as the name of the filter, then click **Finish**. The wizard closes and the action is opened in the Filter Editor.
 - c. In the Filter Editor window, click to place the edit cursor inside the **Content** field.
 - d. Press Ctrl+Spacebar to display the Content Assist box.
 - e. Scroll down in the Content Assist box until you see the phrase the total minutes for month of <a track prepaid minutes>.
 - f. Double-click to select **the total minutes for month of** from the list and add this phrase to the filter content field.
 - g. Select **the track prepaid minutes** from the list.
 - h. Select **is more than** from the list, and then select **<number>**.
 - i. Click the **number** variable and replace the variable with the value 100. This example shows the completed filter phrase:
the total minutes for month of the track prepaid minutes is more than 100
 - j. Click **File > Save**, then close the Filter Editor window.
2. Create the event rule that uses the filter.
 - a. Right-click the **assets** folder, then click **New > Event Rule**.
 - b. In the New Event Rule wizard, enter **added minutes exceeds alert trigger amount** as the name of the event rule.
 - c. Click **Next**.
 - d. In the Select Event panel, click **minutes added to phone** to indicate that the new event rule applies to the selected event.
 - e. Click **Next**.
 - f. In the Select context definition panel, click **System Context**, then click **Next**.
 - g. In the Define the context ID panel, select **the mobile number of the customer** as the context so that the event rule can use the unique mobile phone number to reference the summary business object.
 - h. Click **Finish**. The event rule is opened in the Rule Editor.
 - i. Add the event rule content.
 - 1) Click inside the **Content** field to place the edit cursor in the field.
 - 2) Press Ctrl+Spacebar to open the Content Assist box.
 - 3) Select **if <conditions>** from the list.
 - 4) Continue selecting from the list to add these phrases to the rule:
 - **minutes purchased greater than alert trigger**.
 - **then <actions>**
 - **alert customer suggest new plan**
 - 5) Click the semicolon symbol (;), which indicates the end of the rule content.

The following example shows the completed rule event:



j. Save the event rule and close the Event Editor window.

In the next task, you deploy the tutorial application.

 **Next:** “Task 4: Deploying the application”

Task 4: Deploying the application

In this task, you deploy the tutorial application.

Now that you have completed the event project which defines the tutorial application, you must deploy the event project to the event run time which is running on your system so that you can confirm that the application works.

1. In the Event Explorer view, right-click **MobilePhoneMinutes**, then click **Deploy**.
2. In the Deploy window, check to make sure that the default option **Deploy all assets** is selected, then click **Next**.
3. Select an existing event run time from the Deploy window, then click **Finish**. The event project assets are deployed to the event run time.

If the deployment fails, an error message is displayed in the Problems view. Check that you have defined the assets in the event project as described in the previous tutorial tasks.

You have completed the tutorial. Use the Event Tester widget to test the behavior of the sample event project by submitting events.

Related tasks:

Testing event logic using the Event Tester widget

Tutorial: Invoking a ruleset from an event project

This tutorial shows you how to invoke a ruleset from an event project by creating a business rules data connection. Information about the loan and the customer is evaluated using several rules in the rule project.

In this tutorial, you learn how to define the business rule data connection and then invoke the business rule project ruleset from the event project.

Before you start

In this tutorial, you create an event project, add a business rules data connection, then invoke a ruleset from the event project. The event project defines an event application that tracks loan requests from customers. Each loan request is sent to a ruleset to decide whether the loan is approved.

In this tutorial you complete the following tasks:

- “Task 1: Defining a Business Rules Data Connection” on page 94
- “Task 2: Creating an event rule to use the data connection” on page 96
- “Task 3: Deploying the tutorial application” on page 97

What you will learn

In this tutorial, you learn these skills:

- Define a business rules data connection.
- Deploy the tutorial ruleset.
- Deploy the tutorial event project to the event runtime.
- Define an event to invoke the tutorial ruleset.
- Deploy and test the tutorial application.

This tutorial takes about one hour to complete.

Software requirements and directory structure

This tutorial runs on the version of WebSphere Application Server sample server that includes Rule Execution Server and the event runtime. Start the sample server before running this tutorial.

This tutorial requires the Event Tester Enabler. Make sure that you have installed this feature in one of the following ways:


- By selecting the Event Tester Enabler in IBM Installation Manager at installation time, see [Selecting the features to install](#).
- By installing the product and the sample server using the launchpad, see [Installing the product and the sample server](#).

The files for this tutorial are in the following directory: `odm_install_dir/studio/tutorials/eventinvokeruleset`, which has the following structure:

- `answer`: Answer directory for the tutorial, which contains the completed event project.
- `start`: Start directory for the tutorial, which contains the following shared projects that are required by the tutorial.
 - `invokedloan-xom`: the Java™ execution model for the rule project.
 - `invokedloanruleapp`: the ruleapp project that deploys the ruleset to Rule Execution Server.
 - `invokedloanrules`: the rules project that decides whether the loan is approved.

What you need to know

This tutorial is intended for developers who want to learn how to develop event projects in Decision Server Events. The procedures in the tutorial assume that you are familiar with the Operational Decision Manager Eclipse environment, and that you previously completed the Getting started with event rules tutorial.

 **Start this tutorial:** “Task 1: Defining a Business Rules Data Connection” on page 94.


Tutorial scenario

You can define an event application that invokes a ruleset from a business rules loan project and then stores the loan decision.

To decide whether a loan can be approved for a customer, information about the loan and the customer is evaluated using several rules in the rule project. In this tutorial, you import the rule project that decides whether a loan is approved for a customer. Then you define an event application that invokes this rule project and that stores the loan decision.

To complete the tutorial, you complete these tasks:

- Define a business rule data connection.
- Invoke a business rule project ruleset from an event project.

 **Start this tutorial:** “Task 1: Defining a Business Rules Data Connection” on page 94.

Related information:

“Running the completed tutorial”

You can import and run the tutorial in its completed state. When completed, the tutorial contains an event project that invokes a ruleset.

Running the completed tutorial

You can import and run the tutorial in its completed state. When completed, the tutorial contains an event project that invokes a ruleset.

Start the sample server before beginning this task.

To import the completed project and run the tutorial application:

1. Open the samples console in the en_US (American English) locale.
2. In the Samples and Tutorials perspective, navigate to **Rules and Events Integration > Tutorials > Invoking a ruleset from events > answer**, and then click **Import projects**. The following tutorial projects appear in the Rule Explorer view:
 - The event project `invokingloanevents`, which defines events, event rules and all assets that are required to invoke the ruleset.
 - The rule application project `invokedloanruleapp`, which defines a rule application to deploy the ruleset.
 - The rule project `invokedloanrules` defines the ruleset, which decides whether loans are approved. This project references the project `loan-xom`.
 - The Java project `invokedloan-xom`, which defines an execution object model (XOM) for rule execution.

3. Deploy the tutorial ruleset. If you already deployed the invokedloanrules ruleset because you completed the tutorial tasks, you can skip this step.
 - a. Click to expand the invokedloanruleapp project, then double-click archive.xml to open it in the Rule Application Editor.
 - b. In the Deployment section of the editor, click **Deploy**.
 - c. In the Deploy RuleApp Archive window, click **Increment RuleApp major version**, then click **Next**.
 - d. Confirm the following deployment settings, then click **Finish**.
 - The option **Create a temporary Rule Execution Server Configuration** is selected.
 - The **URL** of the Rule Execution Server is correct.
 - The **port number** indicates the port where Rule Execution Server is running.
 - The login credentials are correct.
 - The option **Deploy XOM of rule projects and archives contained in the RuleApp** is selected.
 - e. Go to the Console view and wait for messages indicating that the rule application was successfully deployed.
4. Deploy the event project to the event runtime.
 - a. Switch to the Event perspective.
 - b. Right-click the event project, then click **Deploy**.
 - c. Make sure that **Deploy all assets** is selected, then click **Next**.
 - d. Click **Use a temporary runtime**, then enter the values for your local event runtime.
 - e. Click **Finish**.
5. Run the tutorial from the event widgets.
 - a. Click **Start > All Programs > IBM > Operational Decision Manager V8.5.1 > Sample Server > Events Tools > Event widgets**.
 - b. Log in using the username and password combination you specified during installation.
 - c. Select the **Event Testing** widget.
 - d. Click **Select an Event Template > Choose Loan Request**, then click **OK**.
 - e. In the **borrower** event section, enter the following customer data:
 - creditScore: 600
 - name: John Doe
 - yearlyIncome: 80000
 - f. In the **loan** event section, enter the following data:
 - amount: 500000
 - duration: 240
 - yearlyInterestRate: 0.5
 - g. Click **Send Event**. The ruleset is executed and the decision results are captured in a report file.
6. Click **Actions**. Confirm that the askForReport action is completed.
 - The askForReport action is persisted. You can confirm the persistence by navigating to the temporary directory for the connectors. The default temporary directory for Windows systems is c:/temp.
 - A report file, report*.xml, is created. The file name includes an identification number that is generated by Rule Execution Server.

7. Log out of the event widgets.

 **Start this tutorial:** “Task 1: Defining a Business Rules Data Connection.”

Related tasks:

Opening the samples console

Related information:

“Tutorial scenario” on page 92

You can define an event application that invokes a ruleset from a business rules loan project and then stores the loan decision.

Task 1: Defining a Business Rules Data Connection

In this task, you define a business rule data connection in an event project.

If you previously imported the answer projects to view the completed tutorial, delete the existing loan projects before importing the start projects.

1. Import the start projects to begin the tutorial tasks.
 - a. Open the samples console in the en_US (American English) locale.
 - b. In the Samples and Tutorials perspective, navigate to **Rules and Events Integration > Tutorials > Invoking a ruleset from events > start**, and then click **Import projects**. The following tutorial projects are added to the Event Explorer view:
 - The Java project `invokedloan-xom`, which defines an execution object model (XOM) for rule execution.
 - The rule project `invokedloanrules` defines the ruleset, which decides whether loans is approved. This project references the project `invokedloan-xom`.
 - The rule application project `invokedloanruleapp`, which defines a rule application to deploy the ruleset.
2. Review the ruleset custom property in the RuleApp project.
 - a. Switch to the Rule perspective.
 - b. In the Rule Explorer view, click to expand the `invokedloanruleapp` project.
 - c. Double-click `archive.xml` to open the file in the RuleApp Editor.
 - d. Click the **Ruleset Archives** tab.
 - e. Select **invokedloanrules** from the Ruleset Archives list.
 - f. In the Ruleset Properties section, review the custom properties table. The property **ruleset.bom.enabled** must be absent or set to true. This property setting is required to successfully invoke the ruleset from an event project.
3. Deploy the ruleset in the RuleApp project. If you previously deployed the ruleset while viewing the completed tutorial, then you can skip this step.
 - a. In the RuleApp editor window, click the **Overview** tab.
 - b. In the Deployment section, click **Deploy**.
 - c. In the Deploy RuleApp window, select **Replace RuleApp version**, then click **Next**.
 - d. Confirm the following deployment settings, then click **Finish**.
 - The option **Create a temporary Rule Execution Server configuration** is selected.
 - The **URL** of the Rule Execution Server is correct.

- The port number part of the URL indicates the port where Rule Execution Server is running.
 - The login credentials are correct.
 - The option **Deploy XOM of rule projects and archives contained in the RuleApp** is selected.
- e. In the Console view, look for messages indicating that the rule application was successfully deployed.
4. Create a loan event project.
 - a. Make sure that you are working in the Events perspective.
 - b. Click **File > New > Event Project**.
 - c. Enter `invokingloanevent` as the event project name, then click **Finish**.
 5. Add a business rules data connection to the event project.
 - a. In the Event Explorer view, right-click the `invokingloanevent` project, then click **New > Data Connection**.
 - b. In the New Connection window, select **Business Rule Data Connection**, then click **Next**.
 - c. Select **invokedloanrules** in the list of rule projects, then click **Next**.
 - d. Make sure that `invokedloanrules` is entered as the name of the data connection, click **Next**.
 - e. In the ruleset details window, click the **RuleApp name** list, then select `invokedloanruleapp` as the RuleApp name. In the **Ruleset name** field, the ruleset name `invokedloanrules` is added automatically.
 - f. In the Testing information for rule application and rule set section, select the **Use a temporary runtime** option. Confirm that the host name, port information, and login credentials are correct, then click **Test connection**. When you see a success message, click **Next** to proceed with the tutorial procedure.
 - g. In the table that shows the ruleset parameters and corresponding business objects that will be generated, review the borrower, loan, and report rows. Note that a business object will be generated for each of these parameters.
 - h. Click **Finish**. The data connection is created and opened in the Connection Editor window.
 6. In the Event Explorer view, click **assets > Business Objects** and explore the new business objects that were generated when the business rules data connection was created.
 - a. Double-click `invokedloanrules_borrower`, then click the **Data mappings** tab. In the Field mappings section, note that the fields `creditScore`, `name`, and `yearlyIncome` each use the IN role and were generated by the `invokedloanrules` data connection.
 - b. Click the **Overview** tab. The `Not constructed nor valid to self-construct` warning is eliminated in the next tutorial task.
 - c. In the Event Explorer view, double-click `invokedloanrules_report`. On the **Overview** tab, in the Fields section, review the fields that use the definition type **Enriched by business rules**. The ruleset is invoked when any of the business objects with this definition type are referenced by an event filter or action. The event runtime computes all the business object fields that correspond to IN parameters from the ruleset, invokes the ruleset, and then, using ruleset results, populates all the business objects that correspond to OUT parameters in the ruleset.

In the next task, you use the generated business objects when you add event objects and actions to the event project.

 **Next:** “Task 2: Creating an event rule to use the data connection”

Related tasks:

Opening the samples console

Task 2: Creating an event rule to use the data connection

In this task, you create an event to populate the event business objects and an action to write an output report when the event project is deployed.

1. Create an event in the invokingloanevents project.
 - a. Make sure that you are working in the Event perspective.
 - b. In the Event Explorer view, right-click the invokingloanevents event project, then click **New > Event**.
 - c. Select **Create a blank event**, then click **Next**.
 - d. In the Name field, enter loanRequest, then click **Next**.
 - e. Make sure that the **Event connection** option is set to **None**, then click **Finish**.
2. Add event objects to the event.
 - a. In the Event Explorer view, right-click the business object invokedloanrules_borrower, then click **Create Event Object from This Business Object**.
 - b. In the Event object window, enter borrower_event in the name field.
 - c. Confirm that the event project name is set to **invokingloanevents**, then select **loanRequest** as the Event to add to, and click **Next**.
 - d. Confirm that all event object fields are selected, then click **Finish**.
 - e. Repeat the steps to create a loan_event event object from the invokedloanrules_loan business object, but instead of selecting all the business object fields, click to clear the selection of the yearlyRepayment field.
3. Create an action in the event project.
 - a. In the Event Explorer view, right-click the invokingloanevents event project, then click **New > Action**.
 - b. Select **Create a blank action**, then click **Next**.
 - c. In the Name field, enter askForReport, then click **Next**.
 - d. Select **File System** in the Event Connection field, then click **Finish**.
 - e. In the Action Editor, click the **Connector** tab, then in the File System Connector Setting section, enter /temp as the folder name, and report*.xml as the file pattern.
 - f. Click **File > Save All** to save your work, then close the editor windows.
4. Add action objects to the action in the event project.
 - a. In the Event Explorer view, right-click the invokedloanrules_report business object, then click **Create Action object from This Business Object**.
 - b. In the Name field, enter report_action.
 - c. Confirm that the event project name is set to **invokingloanevents**, then select **askForReport** as the Action to add to, and click **Next**.
 - d. Confirm that all action object fields are selected, then click **Finish**.

5. Define an event rule using the new event. The if condition part of the event rule checks whether the loan is approved. This condition invokes the ruleset to obtain the loan decision.
 - a. In the Event Explorer view, right-click the `invokingloanevents` event project, then click **New > Event Rule**.
 - b. In the Name field, enter `ReportRequest`, then click **Next**.
 - c. Select the **loanRequest** event, then click **Next**.
 - d. In the Define Context Relationship window, select **the name of the borrower**, then click **Finish**. The `ReportRequest` event rule opens in the Rule Editor.
 - e. In the Content part of the editor window, enter the following rule part:


```
if
the approved of the report is not true
then askForReport ;
```
 - f. Click **File > Save**. Now you can test the event rule using the tutorial application.

In the next task, you run and test the tutorial application.

 **Next:** “Task 3: Deploying the tutorial application”

Task 3: Deploying the tutorial application

In this task, you deploy and run the tutorial application and review the resulting actions and messages.

1. Deploy the event project to the event runtime.
 - a. Switch to the Event perspective.
 - b. Right-click the event project name, then click **Deploy**.
 - c. Make sure that **Deploy all assets** is selected, then click **Next**.
 - d. Click **Use a temporary runtime**, then enter the host name, port, and log in credential values for the event runtime.
 - e. Click **Finish**.
2. Run the tutorial from the event widgets.
 - a. Click **Start > All Programs > IBM > Operational Decision Manager V8.5.1 > Sample Server > Events Tools > Event Widgets**.
 - b. Log in using the username and password credentials that you specified during installation.
 - c. Select the **Event Tester** widget.
 - d. Click **Send Events**.
 - e. Click **Select an Event Template > Choose Loan Request**, then click **OK**.
 - f. In the **borrower** event section, enter the following customer data:
 - `creditScore: 600`
 - `name: John Doe`
 - `yearlyIncome: 80000`
 - g. In the **loan** event section, enter the following data:
 - `amount: 500000`
 - `duration: 240`
 - `yearlyInterestRate: 0.5`

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM United Kingdom Laboratories,
Mail Point 151,
Hursley Park,
Winchester,
Hampshire,
England SO21 2JN

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those

websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM United Kingdom Laboratories,
Mail Point 151,
Hursley Park,
Winchester,
Hampshire,
England SO21 2JN

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or

imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

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



Printed in USA