

Intelligent Event Processor (IEP) Tutorial

Detection of Insider Stock Trading

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

November 2008

Contents

Setting Up the Environment	3
Key Concepts in IEP	6
Detection of Insider Stock Trading	7

Setting Up the Environment

The Intelligent Event Processor (IEP) consists of a design-time side (a set of NetBeans plug-ins) and a runtime side (the IEP Service Engine).

This tutorial assumes that you have installed the IEP design-time files and the runtime file, or that the IEP design-time files and the runtime file are preinstalled.

Setting up the environment consists of the following procedures.

- To Download and Extract the Runtime Projects
- To Start the IEP Service Engine
- To Connect to the IEP Database

To Download and Extract the Runtime Projects

1. In a browser, go to the following URL:
<http://wiki.open-esb.java.net/Wiki.jsp?page=IEPApplicationDeveloper>
2. Locate the Tutorials section and click the Runtime Projects link below the Detection of Insider Stock Trading tutorial.
3. Download the **iep-insider-trading.zip** file to your hard drive.
4. Extract the contents of the **iep-insider-trading.zip** file.

This tutorial refers to the directory where these contents are located as *<lab_root>*.

The runtime projects are called **tableinput** and **datafeed**. These projects are located in the *<lab_root>/iep-insider-trading/projects* directory. You will run these projects later in the tutorial.

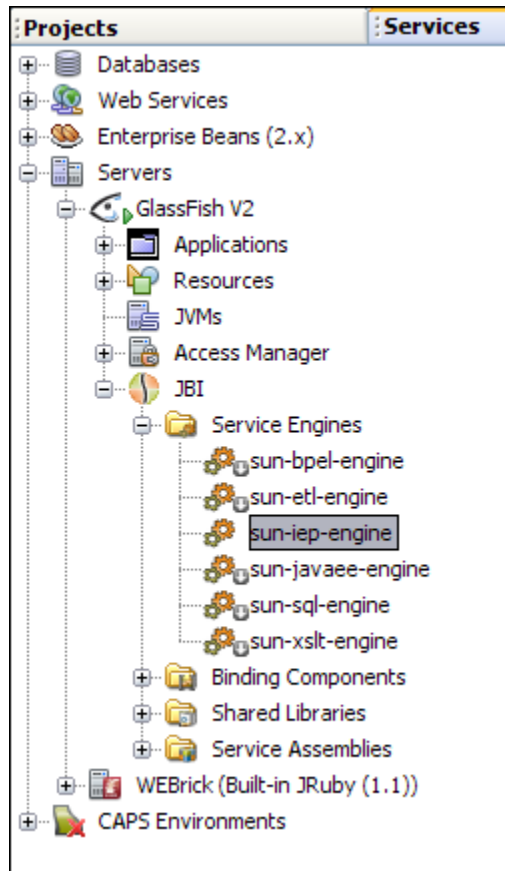
To Start the IEP Service Engine

Before you begin: If you have not already done so, start the NetBeans IDE.

1. In the Services tab, under Servers, right-click GlassFish V2 and start it if necessary.
2. After the components under GlassFish V2 are populated, open JBI > Service Engines to display the service engines that have already been installed.
3. Right-click the **sun-iep-engine** node and select Start.

The IEP Service Engine is started.

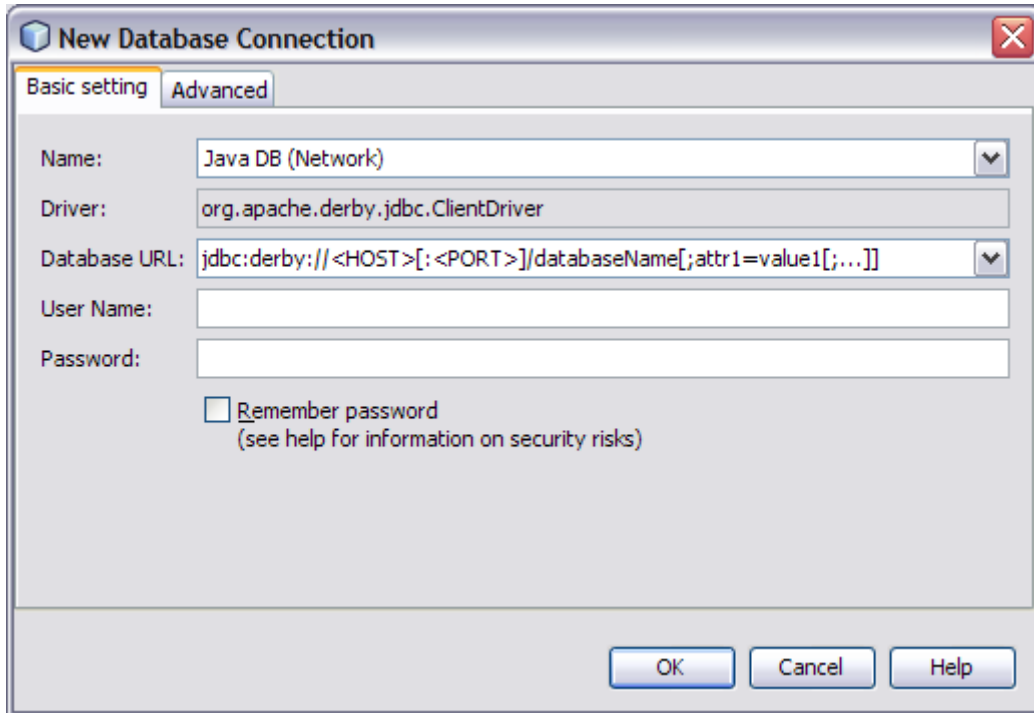
Setting Up the Environment



To Connect to the IEP Database

1. In the Services tab, open Databases > Drivers.
2. Right-click Java DB (Network) and select Connect Using.
The New Database Connection dialog box opens.

Setting Up the Environment



3. In the Database URL field, type **jdbc:derby://localhost:1527/iepseDB**.
4. In the User Name field, type **iepseDB**.
5. In the Password field, type **iepseDB**. The password is masked.
6. Select the Remember Password check box.
7. Click OK.

The connection now appears in the Services tab.

Key Concepts in IEP

Key Concepts in IEP

Table. A *table* is a finite collection of events that belong to a given schema. For example:

Symbol	Price
ADBE	36.50
AMZN	86.50
AMGN	46.50
ADBE	36.60

Stream. A *stream* is a collection (maybe infinite) of timestamped events that belong to a given schema. For example:

Symbol	Price	Timestamp
...
ADBE	36.50	20080214T10:30:02.899-08:00
AMZN	86.50	20080214T10:31:01.674-08:00
AMGN	46.50	20080214T10:31:05.198-08:00
...

Relation (or Sliding Window). A *relation* is a collection of tables that have the same schema and are indexed by time. Relation is also called *sliding window*. For a given time instance t , the table with index t is called the content of the sliding-window at time t .

Example. In the stream of stock transactions above (... , ADBE, AMZN, AMGN, ...), if relation R is defined as “the most recent two stock trades for time t ,” then

- $R(20080214T10:31:01.674-08:00) =$ two latest trades before 10:31:01.674 PST:

Symbol	Price
ADBE	36.50
AMZN	86.50

- $R(20080214T10:31:05.198-08:00) =$ two latest trades before 10:31:05.198 PST:

Symbol	Price
AMZN	86.50
AMGN	46.50

Detection of Insider Stock Training

What You Will Be Doing

Before you begin: Complete the “Setting Up the Environment” section.

In this tutorial, to start, you will be doing the following: (1) Model an input stream of stock trades; and (2) Find the few trades that are “suspicious” (defined as any trade whose price is sufficiently different from the prices in the overall stream, in context). Then, you will: (3) Double-check whether any parties to a suspicious transaction are also named in a table that contains the names of “persons of interest.”

Primary Design Goal: Detect stock trades whose price stands out relative to the stream.

The end result of this goal is an output stream of all suspicious transactions; that is, all trades that have a price significantly different (at least 10 percent above or below) from the stock’s prevailing price. As you achieve this goal, you will learn the difference between streams and relations, how schemas are inherited from one operator to another, and how you can use simple drag-and-drop to construct SQL statements.

Secondary Design Goal: Double-check parties to suspicious trades against a supplied table.

The end result of this goal is an output stream of all suspicious traders; that is, all trades that have a suspicious price that were conducted by people whose names also appear on a table of “persons of interest.” As you achieve this goal, you will learn how to use a static table, and you will see how a single operator’s output can be used for more than one purpose.

Detection of Insider Stock Training

Primary Design Goal: Detect Suspicious Trades

The end result of the primary design goal is an output stream of all suspicious transactions; that is, all trades that have a price significantly different (at least 10 percent above or below) from the stock's prevailing price.

As you achieve this goal, you will learn the difference between streams and relations, how schemas are inherited from one operator to another, and how you can use simple drag-and-drop to construct SQL statements.

Setting Up the Project and Overall Design

- To Create a New IEP Module Project
- To Create a New Intelligent Event Processor
- To Add the Stream Input and Stream Output Operators
- To Add the Intermediate Operators

To Create a New IEP Module Project

Before you begin: If you have not already done so, start the NetBeans IDE.

1. With the NetBeans Projects tab active, select File > New Project.

The New Project dialog box appears.

2. In step 1, make the following choices:

Categories:**SOA**

Projects:.....**Intelligent Event Processing Module**

3. Click Next.

4. In step 2, supply the following information:

Project Name:.....**iep**

Project Location:.....<lab_root>/iep-insider-trading/projects

5. Click Finish.

The project tree displays a new IEP project named **iep** that contains a folder called **Processor Files**.

To Create a New Intelligent Event Processor

1. In the NetBeans Projects tab, under **iep** (the newly created IEP project), right-click **Processor Files** and select New > Intelligent Event Processor.

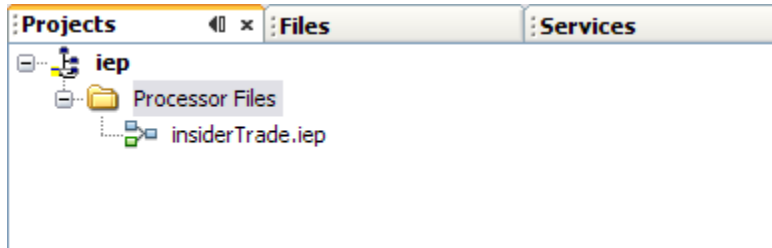
The New Intelligent Event Processor dialog box appears.

2. In the File Name field, type **insiderTrade**.

3. Click Finish.

Detection of Insider Stock Training

In the project tree, the Processor Files folder now contains the file **insiderTrade.iep**, and the IEP Editor opens to allow you to drag IEP operators from the IEP tool palette onto the initially blank design canvas.



To Add the Stream Input and Stream Output Operators

Before you begin: If necessary, select menu Window > Palette to display the IEP tool palette.

1. From the IEP tool palette, under Input, drag a Stream Input operator onto the left side of the design canvas.

This operator defines an *event collector* that allows the event processor to collect events from service engines and binding components.

2. From the IEP tool palette, under Output, drag a Stream Output operator onto the right side of the design canvas.

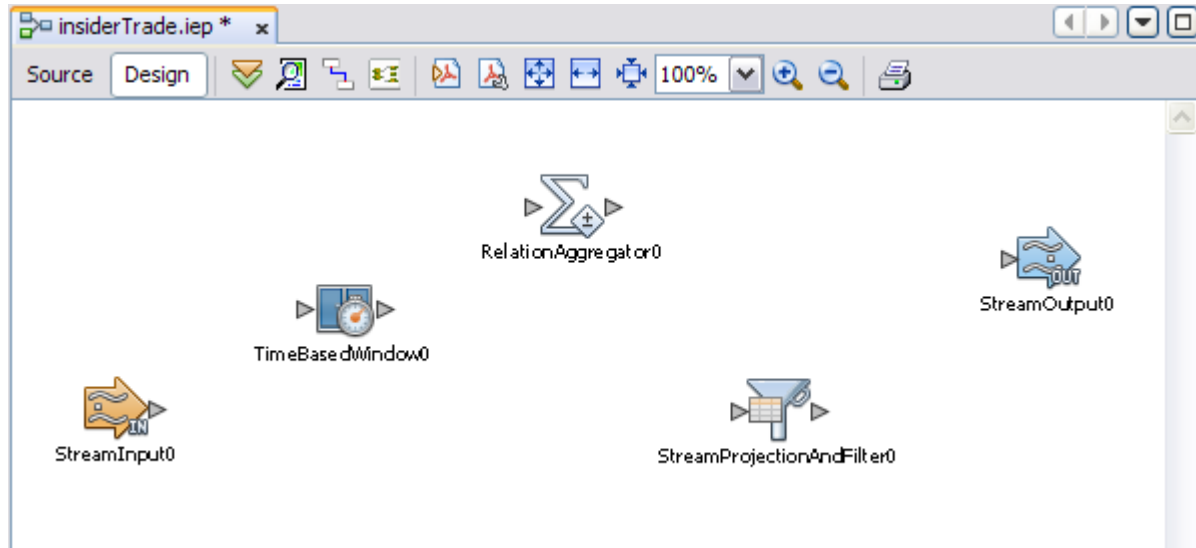
This operator defines an *event notifier* that exposes only the events of interest.

To Add the Intermediate Operators

For now, you will simply add the intermediate operators that perform the filtering and processing functions. You will configure them later. A different and valid work style, not used here, is to configure each operator as you add it.

1. From the IEP tool palette, under Stream Converter, drag a Time Based Window operator onto the design canvas to the right of the StreamInput0 operator.
2. From the IEP tool palette, under Aggregator, drag a Relation Aggregator operator onto the design canvas to the right of the TimeBasedWindow0 operator.
3. From the IEP tool palette, under Correlation and Filter, drag a Stream Projection and Filter operator onto the design canvas to the right of the RelationAggregator0 operator.
4. Position the operators as shown in the following screen capture.

Detection of Insider Stock Training



Connecting and Configuring the Operators

- To Configure the Stream Input Operator: StockTransactions
- To Connect and Configure the Time Based Window Operator: SlidingWindow2Minutes
- To Connect and Configure the Relation Aggregator Operator: RollingAvgPrice
- To Connect and Configure the Stream Projection and Filter Operator: FindSuspiciousTransactions
- To Connect and Configure the Stream Output Operator: SuspiciousTransactions

To Configure the Stream Input Operator: StockTransactions

1. On the design canvas, double-click the StreamInput0 operator.
The property editor opens.
2. In the Name field, change the default value to **StockTransactions**.
3. Click the Add Attribute button three times to add three new blank attributes.
4. Supply names and data types for all four attributes, and supply a size (number of characters) for the two VARCHAR attributes (symbol and trader), as shown in the following table.

Attribute Name	Data Type	Size	Scale	Comment
symbol	VARCHAR	10		
price	DOUBLE			
shares	INTEGER			
trader	VARCHAR	50		

5. Click OK.

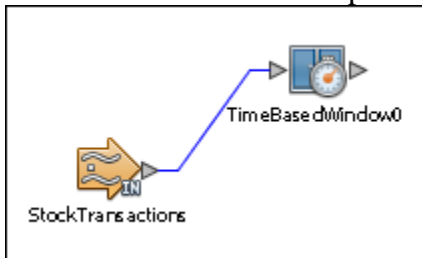
Detection of Insider Stock Training

You have defined an event collector for stock transactions. The four attributes in this event are the stock's ticker symbol, the trading price for that transaction, the number of shares traded in that transaction, and the name of the trader in that transaction.

To Connect and Configure the Time Based Window Operator: SlidingWindow2Minutes

Before you begin: The Stream Input operator, StockTransactions, must already be configured.

1. On the design canvas, click the out-pointing triangle attached to the right of the StockTransactions operator and drag it onto the in-pointing triangle attached to the left of the TimeBasedWindow0 operator.



2. Double-click the TimeBasedWindow0 operator.

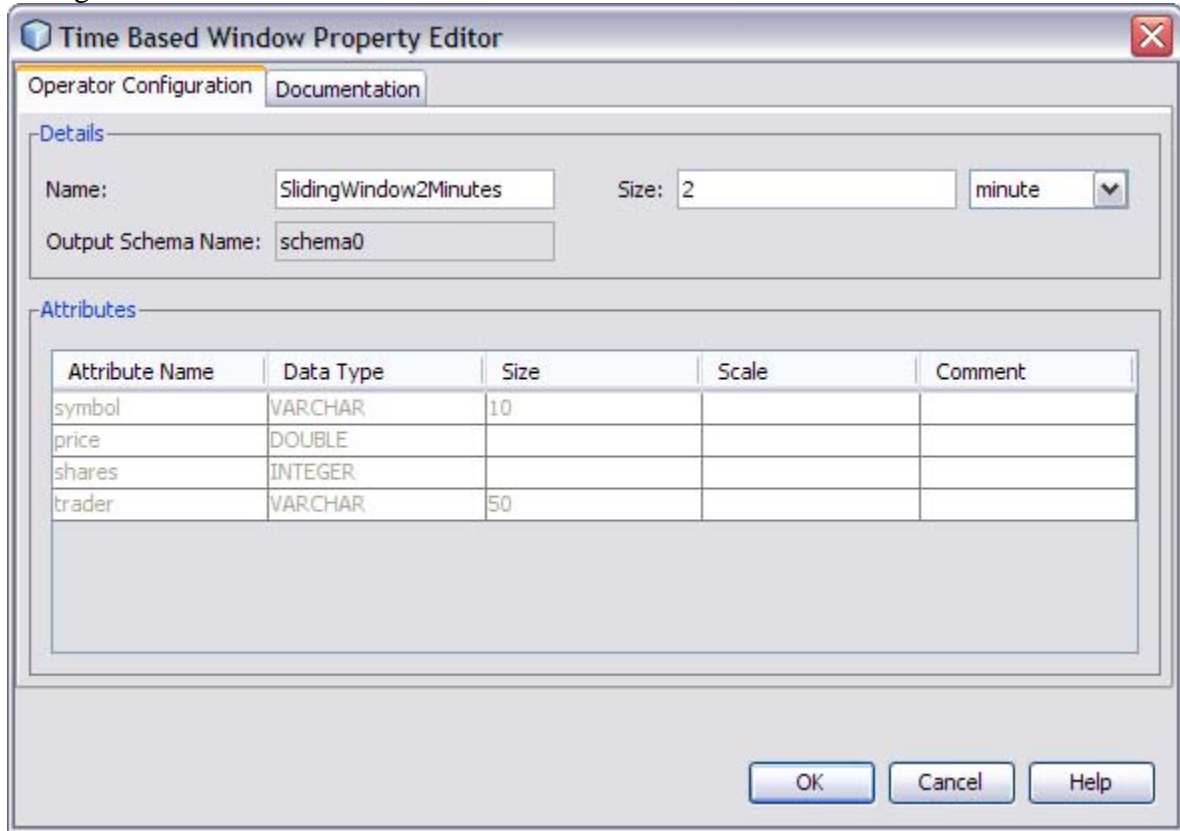
The property editor opens.

Note: Because of the connection, the TimeBasedWindow0 operator inherits the schema of the StockTransactions operator.

3. In the Name field, change the default value to **SlidingWindow2Minutes**.

Detection of Insider Stock Training

4. Change the size and time units to 2 minutes.



5. Click OK.

You have defined a relation in which a set of stock trades transacted over an interval of time (the last two minutes) is focused upon for further examination.

To Connect and Configure the Relation Aggregator Operator: RollingAvgPrice

Before you begin: The Time Based Window operator, SlidingWindow2Minutes, must already be configured.

1. On the design canvas, connect the out-pointing triangle attached to the right of the SlidingWindow2Minutes operator to the in-pointing triangle attached to the left of the RelationAggregator0 operator.
2. Double-click the RelationAggregator0 operator.

The property editor opens.

Note: The RelationAggregator0 operator already contains four items in the Inputs tree, because they were made available by the SlidingWindow2Minutes operator.

3. In the Name field, change the default value to **RollingAvgPrice**.
4. Click the Add Attribute button once to add one new blank attribute.
5. From the Inputs tree, drag the **symbol** attribute into the Expression column of the first attribute and press Enter.

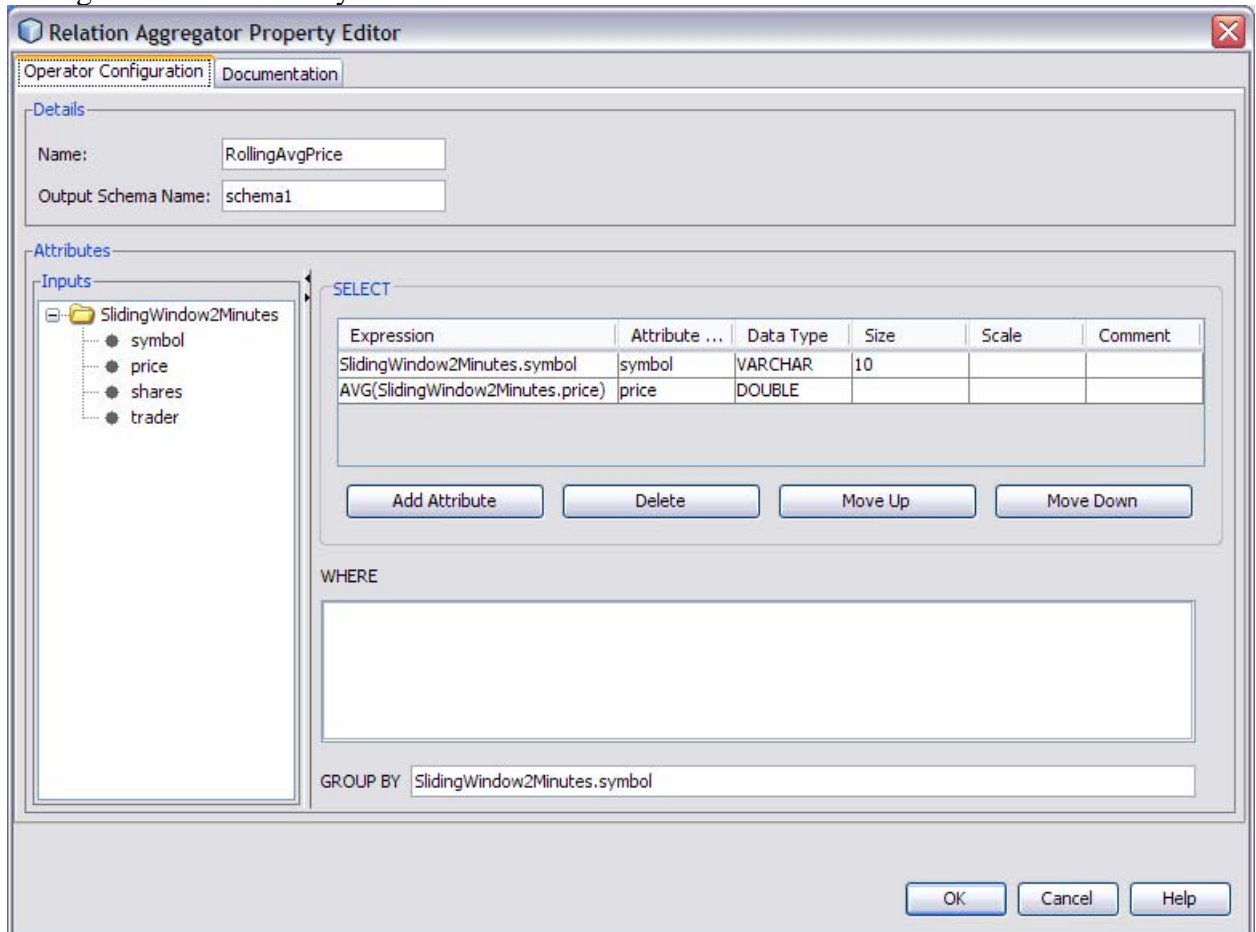
Detection of Insider Stock Training

- From the Inputs tree, drag the **price** attribute into the Expression column of the second attribute and press Enter.
- Enclose the second attribute's expression in the `AVG()` function.

To do this, type the four characters **AVG(** to the left of the expression and then type the single character **)** to the right of the expression. The function takes the average of the **price** attribute of all the events currently contained in the `SlidingWindow2Minutes` operator.

- From the Inputs tree, drag the **symbol** attribute into the GROUP BY field.

The input stream can include more than one symbol. This step enables you to obtain the average across the same symbol.



- Click OK.

To Connect and Configure the Stream Projection and Filter Operator: FindSuspiciousTransactions

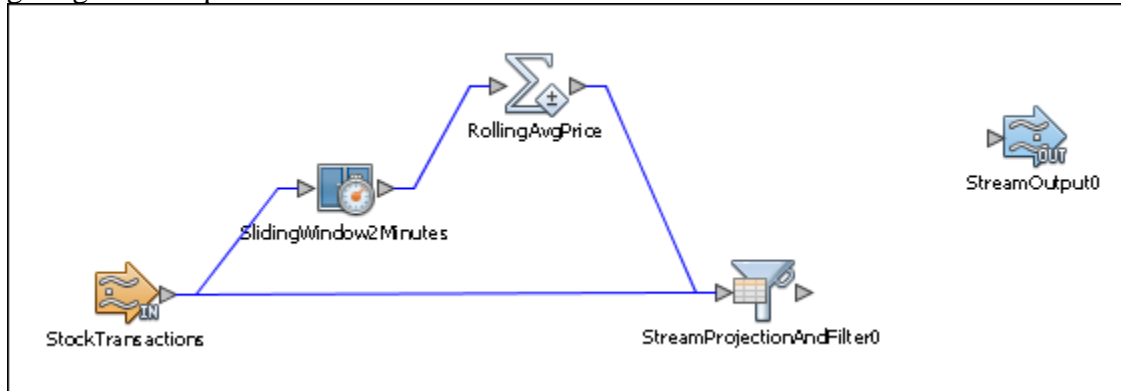
Before you begin: The Relation Aggregator operator, `RollingAvgPrice`, must already be configured.

- On the design canvas, connect the `RollingAvgPrice` operator to the `StreamProjectionAndFilter0` operator.

Detection of Insider Stock Training

The Stream Projection and Filter operator enables you to join a stream with multiple relations and tables, in order to compute new events or to filter existing events based on specified conditions.

2. Also connect the StockTransactions operator to the StreamProjectionAndFilter0 operator, giving it two inputs.



3. Double-click the StreamProjectionAndFilter0 operator.
The property editor opens.
4. In the Name field, change the default value to **FindSuspiciousTransactions**.
5. Click the Add Attribute button three times to add three new blank attributes.
6. From the Inputs tree, under StockTransactions, drag each of the four attributes (**symbol**, **price**, **shares**, **trader**) into each of the four blank attribute rows.
7. From the Inputs tree, drag RollingAvgPrice into the FROM field after the existing text.
8. Either using drag-and-drop and key-ins, or by copying-and-pasting from the following text, construct the WHERE clause so as to include trades where the StockTransactions symbol matches the RollingAvgPrice symbol and the transaction price is significantly different (more than 10 percent above or less than 10 percent below) from the rolling average price.

The SELECT, FROM, and WHERE fields should look like this:

SELECT:

```
StockTransactions.symbol  
StockTransactions.price  
StockTransactions.shares  
StockTransactions.trader
```

FROM:

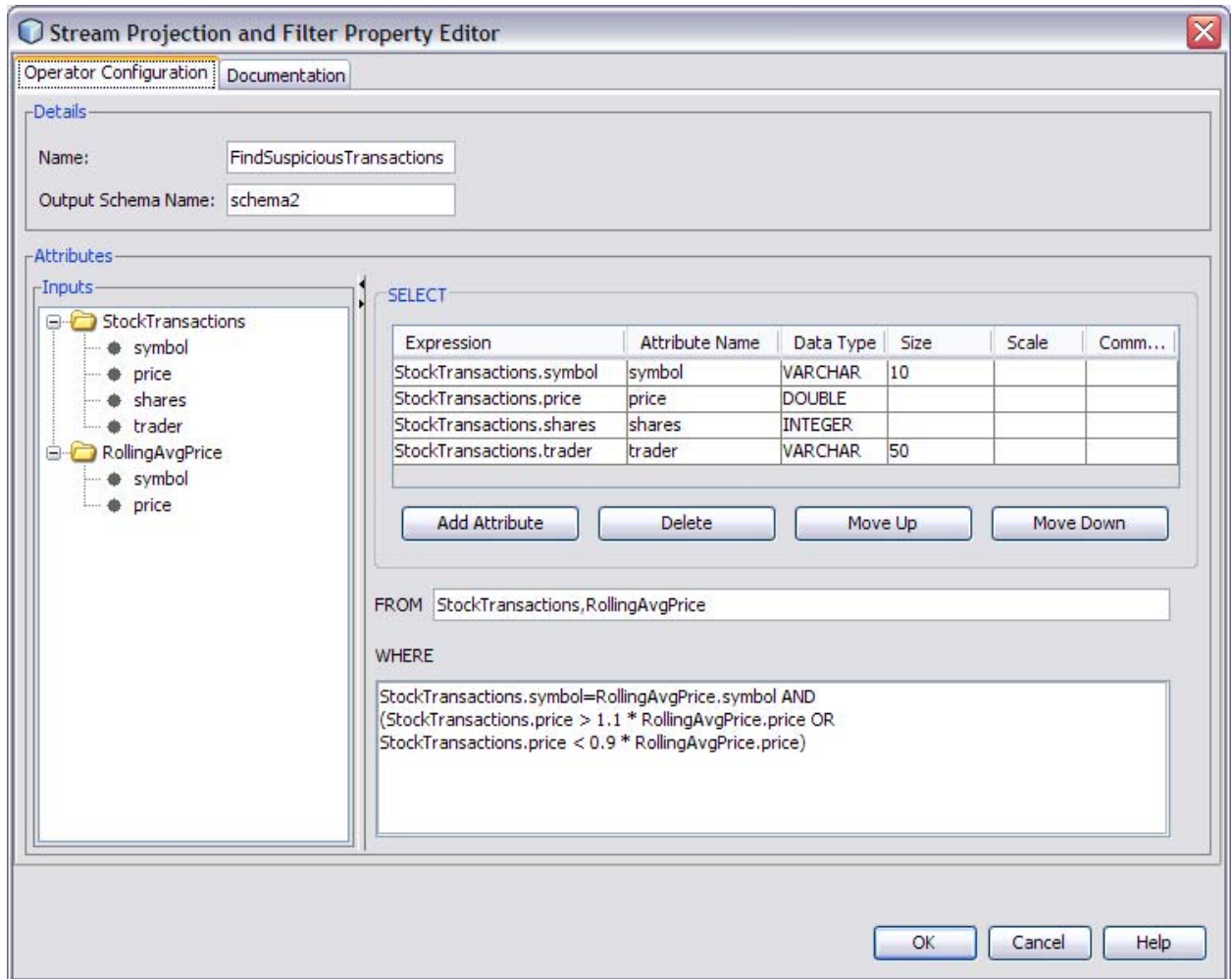
```
StockTransactions, RollingAvgPrice
```

WHERE:

```
StockTransactions.symbol=RollingAvgPrice.symbol AND  
(StockTransactions.price > 1.1 * RollingAvgPrice.price OR  
StockTransactions.price < 0.9 * RollingAvgPrice.price)
```

Detection of Insider Stock Training

Important! Notice the parentheses enclosing the OR expression in the WHERE field.



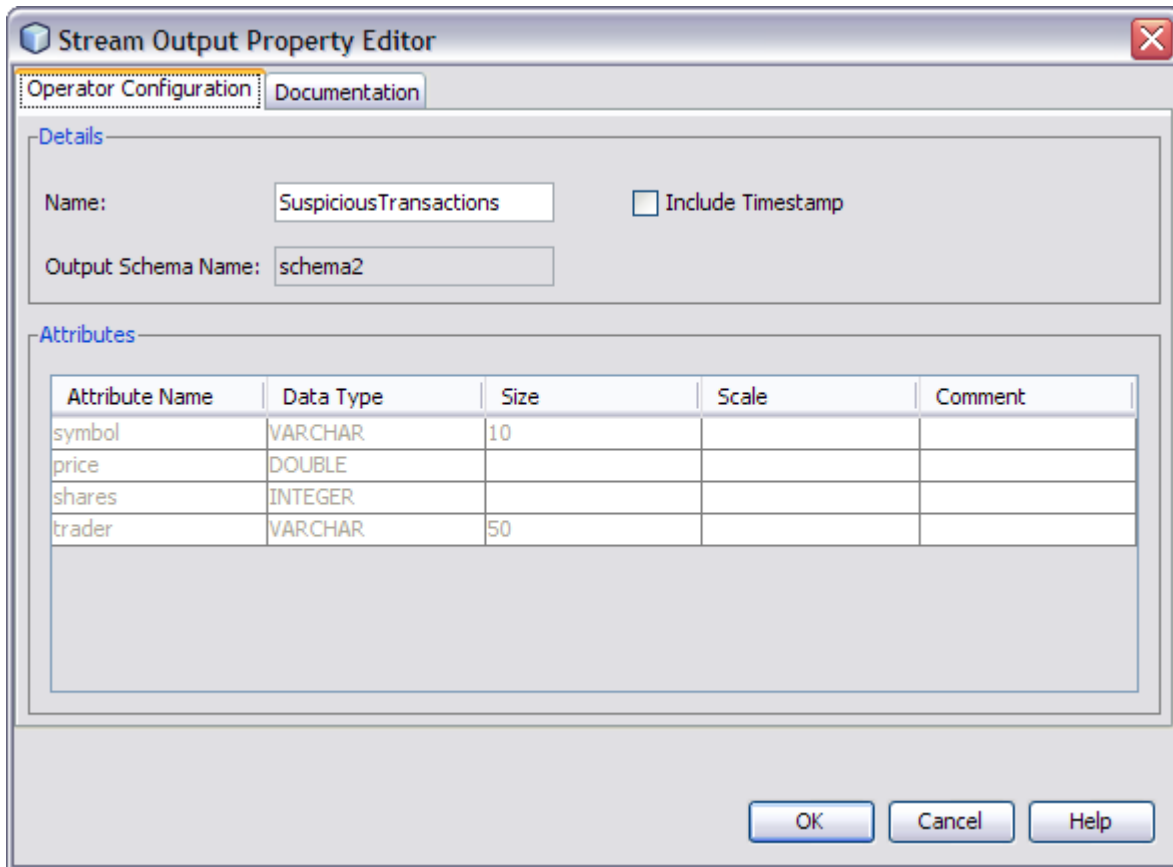
9. Click OK.

To Connect and Configure the Stream Output Operator: SuspiciousTransactions

1. On the design canvas, connect the FindSuspiciousTransactions operator to the StreamOutput0 operator.
2. Double-click the StreamOutput0 operator.
The property editor opens.

Detection of Insider Stock Training

3. In the Name field, change the default value to **SuspiciousTransactions**.



The image shows a 'Stream Output Property Editor' dialog box with two tabs: 'Operator Configuration' and 'Documentation'. The 'Operator Configuration' tab is active. It has two sections: 'Details' and 'Attributes'. In the 'Details' section, the 'Name' field is set to 'SuspiciousTransactions', the 'Output Schema Name' is 'schema2', and the 'Include Timestamp' checkbox is unchecked. The 'Attributes' section contains a table with the following data:

Attribute Name	Data Type	Size	Scale	Comment
symbol	VARCHAR	10		
price	DOUBLE			
shares	INTEGER			
trader	VARCHAR	50		

At the bottom of the dialog are three buttons: 'OK', 'Cancel', and 'Help'.

4. Click OK.
5. Save your work to date.

The WSDL file for the IEP Module project (**insiderTrade.wsdl**) is automatically generated. However, the WSDL file does not appear immediately in the Projects tab. You will edit the WSDL file in a later procedure.

You have achieved the primary design goal of this tutorial. You have created an Intelligent Event Processor, **insiderTrade.iep**, that monitors a stream of stock transactions and detects those with prices that fall outside the norm.

Detection of Insider Stock Training

Secondary Design Goal: Match Traders Against Persons of Interest Table

The end result of the secondary design goal is an output stream of all suspicious traders; that is, all trades that have a suspicious price that were conducted by people whose names also appear on a table of “persons of interest” from an external source.

As you achieve this goal, you will learn how to use a static table, and you will see how a single operator’s output can be used for more than one purpose.

Setting Up the Overall Design

- To Add the Table Input and Stream Output Operators
- To Add the Intermediate Operator

To Add the Table Input and Stream Output Operators

1. From the tool palette, under Input, drag a Table Input operator onto the lower left side of the design canvas.

This operator will be populated with static data that is read from an external source.

2. From the tool palette, under Output, drag a Stream Output operator onto the lower right side of the design canvas.

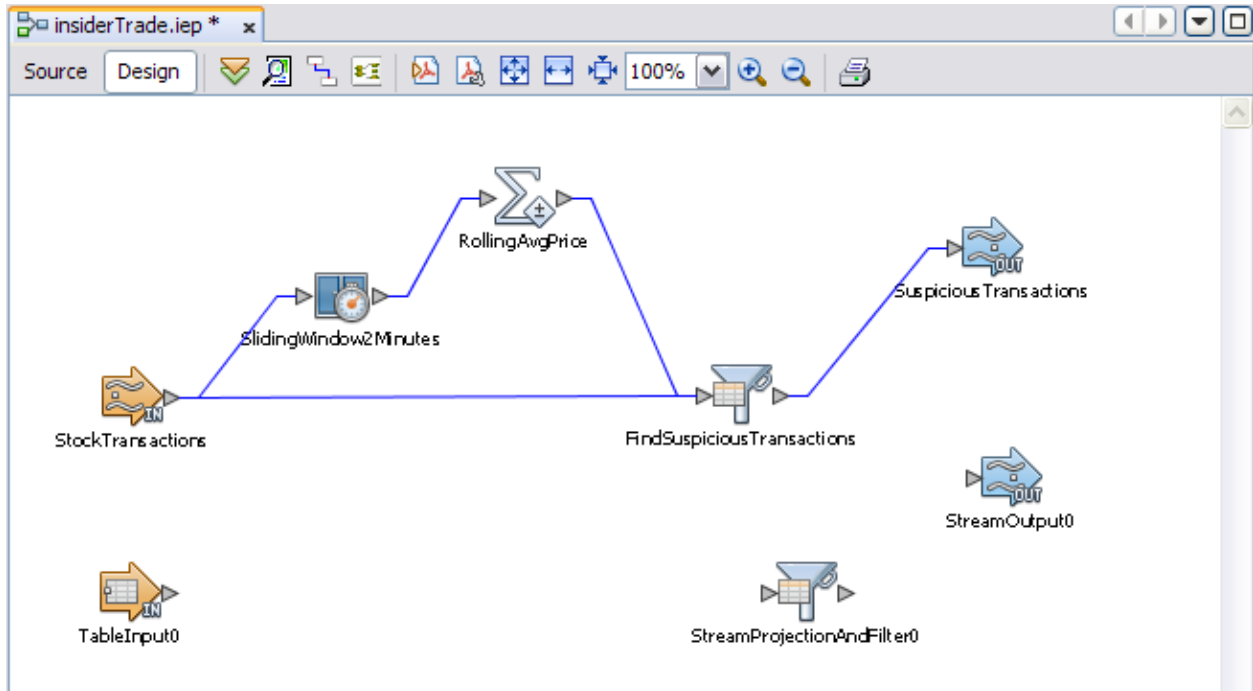
This operator defines an event notifier that exposes only the events of interest.

To Add the Intermediate Operator

For now, you will simply add an intermediate operator that performs the filtering and processing functions. You will connect and configure the operator in a later procedure.

1. From the tool palette, under Correlation and Filter, drag a Stream Projection and Filter operator onto the design canvas in between the TableInput0 operator and the StreamOutput0 operator.
2. Position the operators as shown in the following screen capture.

Detection of Insider Stock Training



Connecting and Configuring the Operators

- To Configure the Table Input Operator: PersonsOfInterest
- To Connect and Configure the Stream Projection and Filter Operator: FindTransactionsOfInterest
- To Connect and Configure the Stream Output Operator: TransactionsOfInterest

To Configure the Table Input Operator: PersonsOfInterest

1. On the design canvas, double-click the TableInput0 operator.
The property editor opens.
2. In the Name field, change the default value to **PersonsOfInterest**.
3. Also change the Global ID to **PersonsOfInterest**.

Detection of Insider Stock Training

4. In the attribute row, specify the following values for the attribute name, data type, and size: name, VARCHAR, and 50.

The screenshot shows the 'Table Input Property Editor' dialog box. It has two tabs: 'Operator Configuration' (selected) and 'Documentation'. Under the 'Details' section, there are fields for 'Name' (PersonsOfInterest), 'Global ID' (PersonsOfInterest), and 'Output Schema Name' (schema3). There is also a checkbox for 'Do Not Create Table' which is unchecked. The 'Attributes' section contains a table with the following data:

Attribute Name	Data Type	Size	Scale	Comment
name	VARCHAR	50		

Below the table are buttons for 'Add Attribute', 'Delete', 'Move Up', and 'Move Down'. At the bottom right of the dialog are 'OK', 'Cancel', and 'Help' buttons.

5. Click OK.

You have defined a static table that will hold data that is read in from an external source.

To Connect and Configure the Stream Projection and Filter Operator: FindTransactionsOfInterest

1. On the design canvas, connect the PersonsOfInterest operator to the StreamProjectionAndFilter0 operator.
2. Also connect the FindSuspiciousTransactions operator to the StreamProjectionAndFilter0 operator, giving it two inputs.
3. Double-click the StreamProjectionAndFilter0 operator.
The property editor opens.
4. In the Name field, change the default value to **FindTransactionsOfInterest**.
5. Click the Add Attribute button three times to add three new blank attributes.
6. From the Inputs tree, under FindSuspiciousTransactions, drag each of the four attributes (**symbol**, **price**, **shares**, **trader**) into each of the four blank attribute rows.

Detection of Insider Stock Training

- From the Inputs tree, drag PersonsOfInterest into the FROM field after the existing text.
- Either using drag-and-drop and a key-in, or by copying-and-pasting from the following text, construct the WHERE clause so as to include only those transactions where the “trader” value of FindSuspiciousTransactions matches the “name” value of PersonsOfInterest.

The SELECT, FROM, and WHERE fields should look like this:

SELECT:

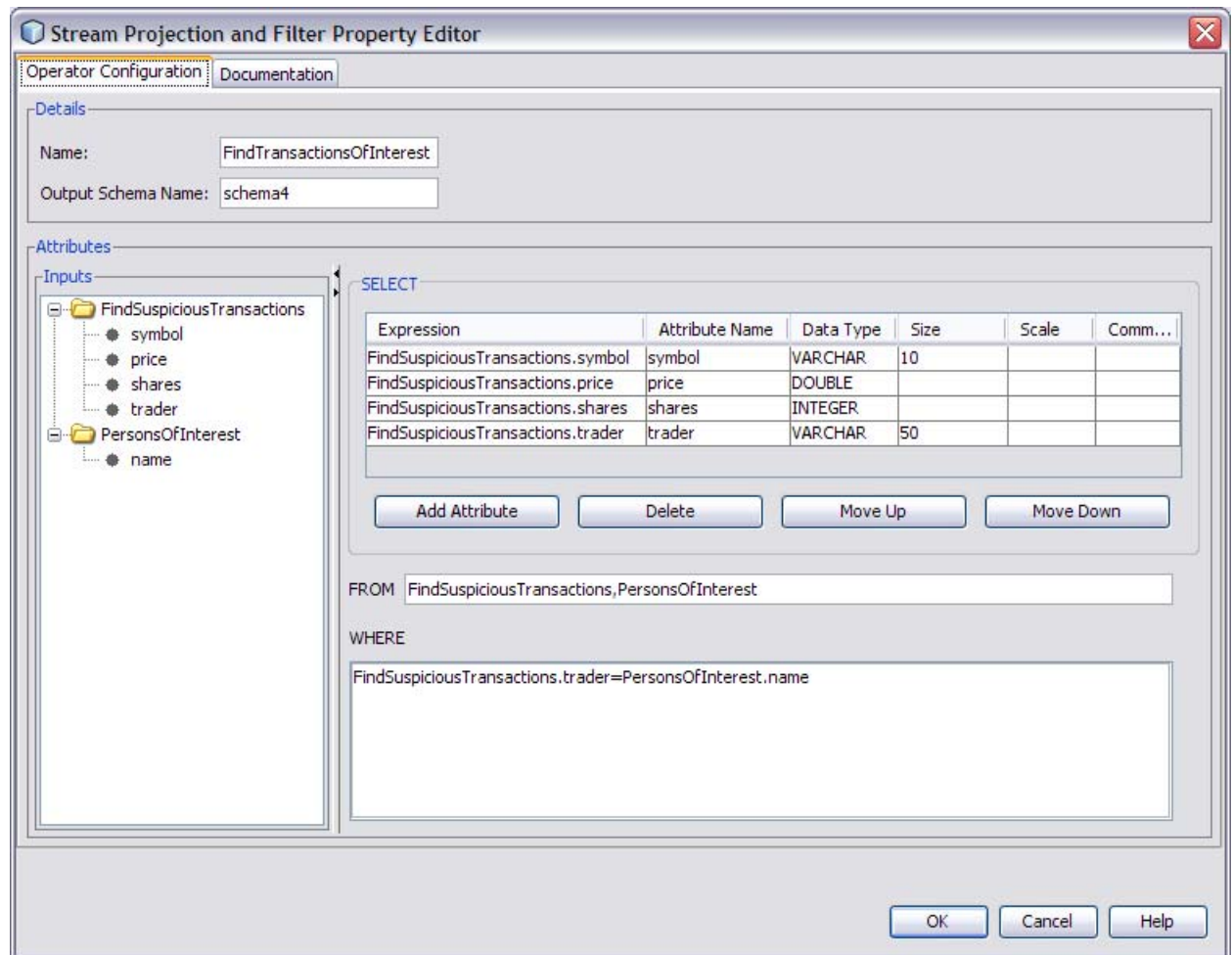
```
FindSuspiciousTransactions.symbol  
FindSuspiciousTransactions.price  
FindSuspiciousTransactions.shares  
FindSuspiciousTransactions.trader
```

FROM:

```
FindSuspiciousTransactions,PersonsOfInterest
```

WHERE:

```
FindSuspiciousTransactions.trader=PersonsOfInterest.name
```



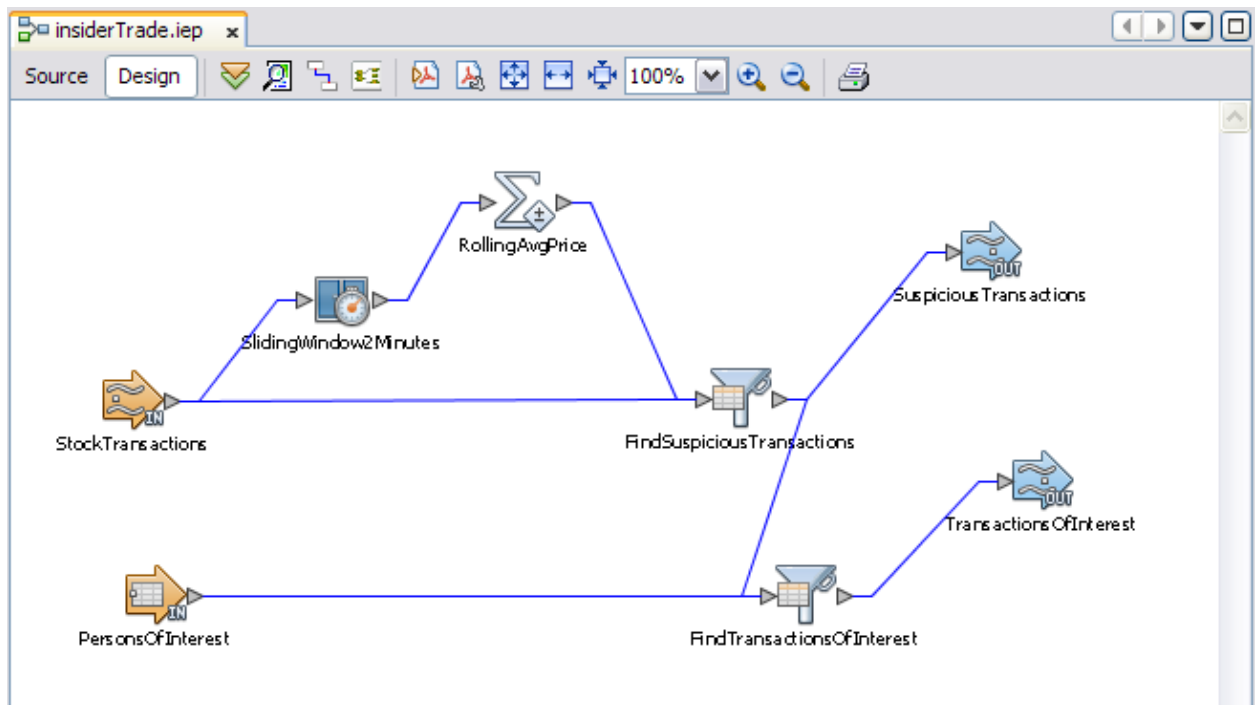
- Click OK.

Detection of Insider Stock Training

To Connect and Configure the Stream Output Operator: TransactionsOfInterest

1. On the design canvas, connect the FindTransactionsOfInterest operator to the StreamOutput0 operator.
2. Double-click the StreamOutput0 operator.
The property editor opens.
3. In the Name field, change the default value to **TransactionsOfInterest**.
4. Select the Include Timestamp check box.
5. Click OK.
6. Save your work to date.
7. Press Alt+Shift+F9 or click the Validate XML button to validate **insiderTrade.iep**.

You have achieved the primary and secondary design goals of this tutorial. You have created an event processor, **insiderTrade.iep**, that monitors a stream of stock transactions, detects those with prices that fall outside the norm, and compares the name of the traders transacting the trade against a table of “persons of interest.”



Detection of Insider Stock Training

Running the Event Processor

The end result of this part of the tutorial is to run the event processor against sample input data and see how it works.

Setting Up for Runtime

- To Edit the Output Directories by Customizing the WSDL File
- To Create the Composite Application Project

To Edit the Output Directories by Customizing the WSDL File

1. In the NetBeans project tree, under iep > Processor Files, open the **insiderTrade.wsdl** file.
2. When the WSDL editor appears, change to the Source view.
3. If the line numbers do not appear in the Source view, then choose View > Show Line Numbers.
4. Scroll down to the end of the WSDL file. Two service elements appear.
5. Within each of the two service elements, change the value of the **fileDirectory** attribute to **<lab_root>/iep-insider-trading/projects**. Be sure to use forward slashes, even on Windows systems.

```
181 <service name="OutputService_SuspiciousTransactions">
182   <port name="OutputPort_SuspiciousTransactions" binding="tns:OutputBinding_SuspiciousTransactions">
183     <file:address fileDirectory="C:/iep-insider-trading/projects"/>
184   </port>
185 </service>
186 <service name="OutputService_TransactionsOfInterest">
187   <port name="OutputPort_TransactionsOfInterest" binding="tns:OutputBinding_TransactionsOfInterest">
188     <file:address fileDirectory="C:/iep-insider-trading/projects"/>
189   </port>
190 </service>
191 <!-- end of file output binding and service -->
192 </definitions>
193
```

6. Save your changes.

To Create the Composite Application Project

1. With the NetBeans Projects tab active, select File > New Project.
The New Project dialog box appears.
2. In step 1, make the following choices:
Categories:**SOA**
Projects:.....**Composite Application**
3. Click Next.
4. In step 2, supply the following information:

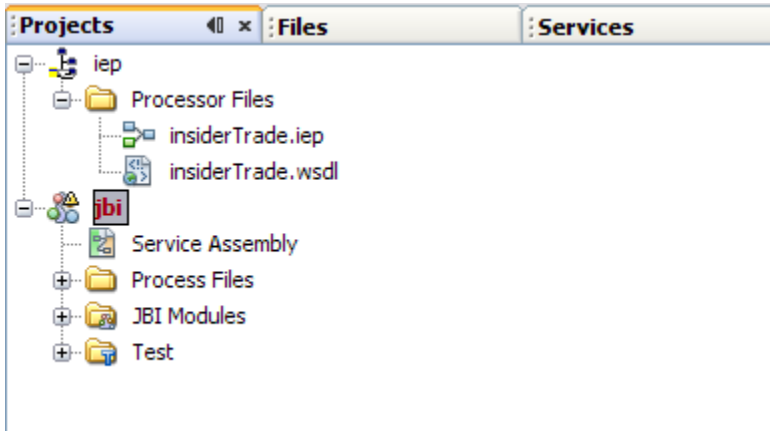
Detection of Insider Stock Training

Project Name:.....**jbi**

Project Location:<lab_root>/iep-insider-trading/projects

5. Click Finish.

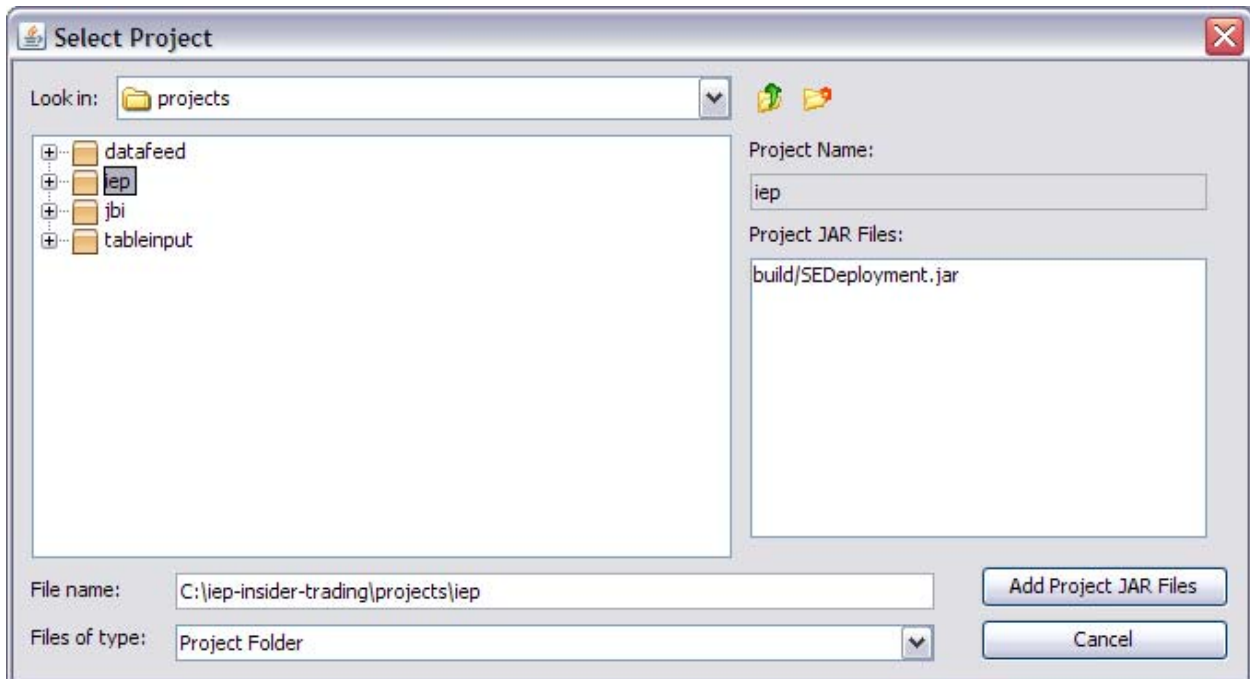
The project tree displays a new Composite Application project named **jbi** that contains several folders. In addition, the CASA Editor appears.



6. Right-click the JBI Modules folder and select Add JBI Module.

The Select Project dialog box appears.

7. Navigate to the <lab_root>/iep-insider-trading/projects folder and select the **iep** project.



8. Click Add Project JAR Files.

The **iep** project is added to the JBI Modules area of the CASA Editor.

Detection of Insider Stock Training

9. Save your changes.
10. Right-click the **jbi** project and select Build.

The connections in the event process are added to the CASA Editor. The connections are based on the default bindings generated by IEP. The input is sent to the event process by HTTP. The output is sent to two files.

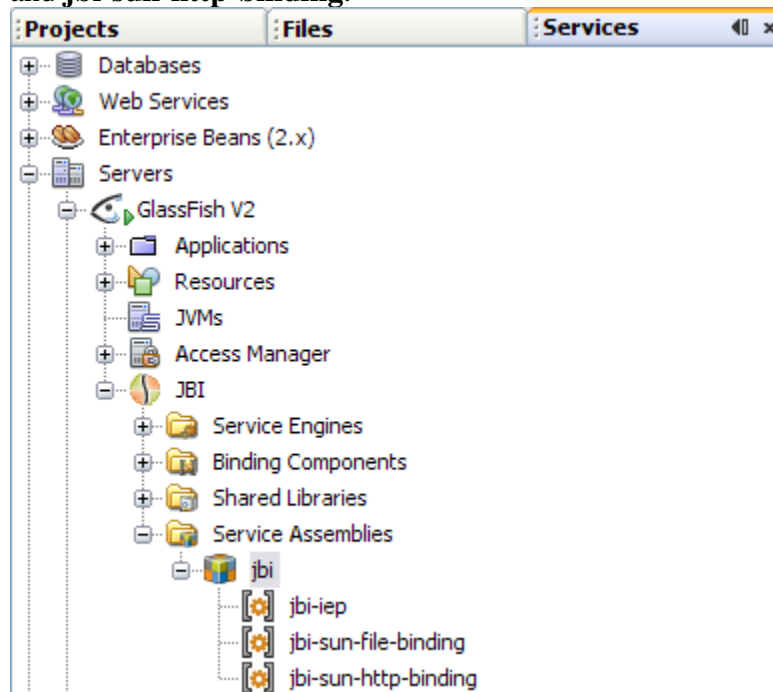
11. Save your changes.

Running the Project

- To Deploy the Composite Application Project
- To Verify That the Persons of Interest Database Table Is Unpopulated
- To Populate the Persons of Interest Database Table and Send Events
- To Check the Results

To Deploy the Composite Application Project

1. With the NetBeans Projects tab active, right-click the **jbi** project and select Deploy.
2. Wait until the BUILD SUCCESSFUL message appears in the Output window.
3. Go to the Services tab and expand Servers > GlassFish V2 > JBI > Service Assemblies > jbi.
4. Notice that the following service units have been deployed: **jbi-iep**, **jbi-sun-file-binding**, and **jbi-sun-http-binding**.



To Verify That the Persons of Interest Database Table Is Unpopulated

1. In the Services tab, expand Databases > jdbc:derby://localhost:1527/iepseDB.

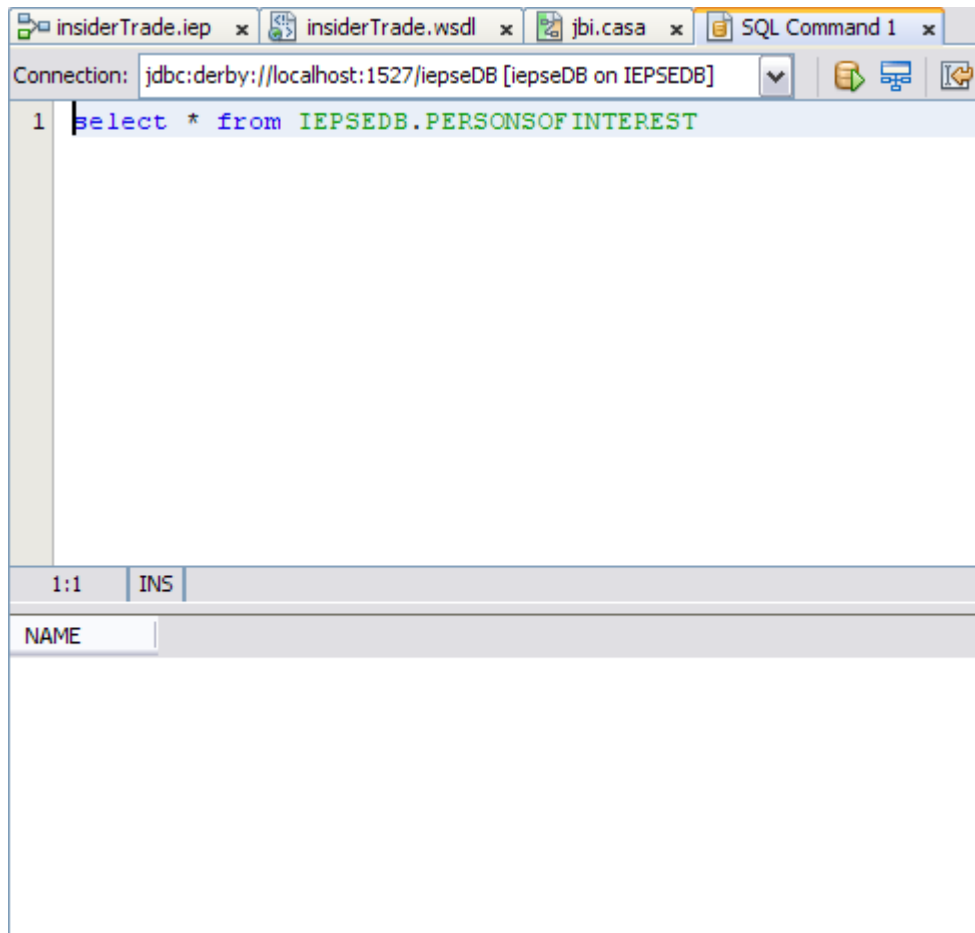
Detection of Insider Stock Training

If the **jdbc:derby://localhost:1527/iepseDB** node does not have any subnodes, then right-click the **jdbc:derby://localhost:1527/iepseDB** node and select Connect.

2. In the Tables folder, right-click the **PERSONSOFINTEREST** table and select View Data.

If the **PERSONSOFINTEREST** table does not appear, then right-click the Tables folder and select Refresh.

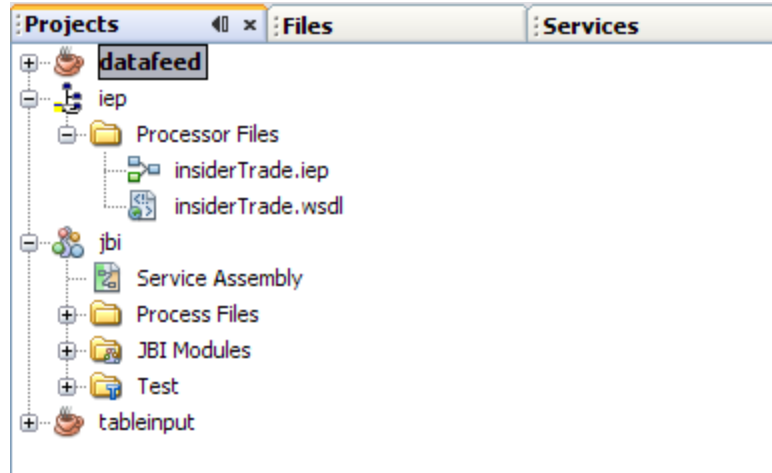
3. In the SQL Command 1 tab, notice that the **select *** command returns no values.



To Populate the Persons of Interest Database Table and Send Events

1. Use File > Open Project to open the **tableinput** project in the `<lab_root>/iep-insider-trading/projects` directory.
2. Use File > Open Project to open the **datafeed** project in the `<lab_root>/iep-insider-trading/projects` directory.

Detection of Insider Stock Training



3. In the NetBeans project tree, right-click the **tableinput** project and select Run.
4. Wait until the BUILD SUCCESSFUL message appears in the Output window.
The Persons of Interest database table is populated.
5. To verify that the table is populated, click the Run SQL button in the SQL Command 1 tab.
6. In the NetBeans project tree, right-click the **datafeed** project and select Run.
As you can see in the Output window, a large number of events is sent into the data stream. Some of the events have suspicious prices, but most do not. Some of the events with suspicious prices may have been transacted by a “person of interest.”

To Check the Results

1. Go to the `<lab_root>/iep-insider-trading/projects` directory.
2. Open the **SuspiciousTransactions.txt** file to see the results.

```
<msgns:SuspiciousTransactions_MsgObj xmlns:msgns="insiderTrade_iep">
<symbol>XYZ</symbol>
<price>19.75</price>
<shares>200000</shares>
<trader>CatchMeIfYouCan</trader>
</msgns:SuspiciousTransactions_MsgObj>
```

```
<msgns:SuspiciousTransactions_MsgObj xmlns:msgns="insiderTrade_iep">
<symbol>XYZ</symbol>
<price>19.79</price>
<shares>150000</shares>
<trader>YouCannotTouchMe</trader>
</msgns:SuspiciousTransactions_MsgObj>
```

Detection of Insider Stock Training

3. Open the **TransactionsOfInterest.txt** file to see the results.

Notice that the **TransactionsOfInterest.txt** file contains a subset of the transactions in the **SuspiciousTransactions.txt** file.

```
<msgns:TransactionsOfInterest_MsgObj xmlns:msgns="insiderTrade_iep">
<symbol>XYZ</symbol>
<price>19.75</price>
<shares>200000</shares>
<trader>CatchMeIfYouCan</trader>
<Timestamp>2008-11-05T12:16:02.838-08:00</Timestamp>
</msgns:TransactionsOfInterest_MsgObj>
```

Detection of Insider Stock Training

Recap: Detection of Insider Trading

In this tutorial, you built and ran a simple Intelligent Event Processing composite application. Along the way, you learned some key IEP concepts and operators.

Key IEP Concepts

- **Table.** A finite collection of events that belong to the same schema. The schema is also called the table's schema.
- **Stream.** A collection (maybe infinite) of events that belongs to the same schema, and have timestamps. The schema is also called the stream's schema.
- **Relation (or Sliding-Window).** A collection of tables that have the same schema, and are indexed by time. The schema is also called the relation's schema. Relation is also called sliding-window. For a given time instance t , the table with index t is called the *content* of the sliding-window at time t .

IEP Operators Used

- **Stream Input.** This operator defines an event collector so that the event processor can collect events from other service engines or binding components.
- **Table Input.** This operator defines a table so that the event processor can correlate it with events from event collectors. The event processor never writes to this kind of table.
- **Stream Output.** This operator defines an event notifier so that the event processor can deliver events to other service-engines/binding-components.

In this tutorial, you used two different Stream Output operators. One was used to output a stream of trades suspicious by virtue of an unusual price. The other was used to output a stream of trader names suspicious for both having made an unusual trade and also appearing in the persons-of-interest table.

- **Time Based Window.** This operator takes a stream as input and computes a sliding window as output. The sliding window's size can be specified as time-period. For example, a sliding-window of size 2 minutes keeps the latest 2 minutes' worth of events relative to any time instance t . As t changes, so does the content of the sliding window.
- **Relation Aggregator.** This operator takes a sliding window U as input and computes a sliding window V as output. V holds the statistical summary of the content of U . For example, given the latest 2-minute window of transactions of a stock, it can compute the minimum, average, and maximum of the stock price of those transactions. As time passes by, the statistical summary changes.

In this tutorial, you used a Relation Aggregator that computed the average price of a particular stock received from the Time Based Window.

- **Stream Projection and Filter.** This operator takes a stream U along with zero or more tables as input, and computes a stream V as output. It lets you specify filtering criteria, and then it filters events in U and joins the filtered result with input tables. For example, given a stream of transactions of a stock, the operator could compute a new stream that holds only those transactions whose price is greater than \$50.

Detection of Insider Stock Training

In this tutorial, you used two different Stream Projection and Filter operators. In the first, you compared the price of a particular trade against a rolling two-minute average price and retained only the trades that were unusual by more by 10 percent. In the second, you compared the party to a particular trade against a table containing a list of names, and retained only those trader names that appeared in both.