IBM Opportunity Detection

Version 9 Release 1
October 25, 2013

User's Guide

Note
Before using this information and the product it supports, read the information in "Notices" on page 43.

This edition applies to version 9, release 1, modification 0 of IBM Opportunity Detection (product number 5725-D16) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Chapter 1. About IBM Opportunity Detection

IBM® Opportunity Detection enables you to look for specified customer behaviors and patterns in your customer data and respond to them. You define the transactions and patterns that Opportunity Detection looks for, and the data that is written to the database when those criteria are met.

You use Opportunity Detection components to build trigger systems in workspaces where you apply your marketing business logic to your transaction and profile data. When you run a trigger system, it processes streams of data from your transaction and profile data feeds.

Batch and real time modes

Opportunity Detection has two modes: batch and real time.

- When installed in a stand-alone configuration, Opportunity Detection processes streams of data in batches.
- When Opportunity Detection is integrated with IBM Interact, it is also possible to apply Pattern component logic to data in real time.

  To learn more about this integration, see the IBM Interact Administrator's and User's Guides.

About trigger systems

A trigger system is comprised of components that you configure. Some components send events that activate other components, while others simply perform operations on data and make the results available to other components. A set of components in a system is said to trigger when it writes its outcome.

Trigger systems can be quite complex and sophisticated, but at a minimum every trigger system requires the following.

- A data source

  Data sources must include transaction data and and can also include static customer profile data.

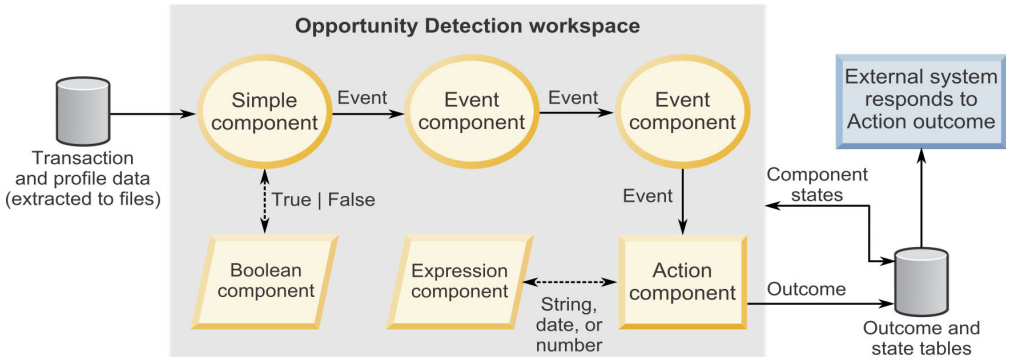
  Data sources are described in the *IBM Opportunity Detection Administrator's Guide*.
- A Simple component

  Looks for specified criteria based on a single audience level in a single transaction data source.
- An Action component

  Writes an outcome to a database table, and this is available for use by external systems.

For example, you could test transaction data for printer purchases, and if a customer purchases a printer but does not buy a printer cartridge within three months after the purchase, your could send a sale coupon for a printer cartridge to each customer who purchases a printer. The outcome your trigger system places in the database could be the string "Send printer cartridge coupon" plus a customer ID. An external system could then respond to this data.

The following diagram illustrates a basic trigger system.



Chapter 2. Opportunity Detection roles and permissions

The permissions assigned to users in Opportunity Detection determine what areas of the application they can access and the actions they can perform.

Reference: Permissions for Opportunity Detection

The following table describes permissions that you can assign to roles in Opportunity Detection.

All permissions that have the **Not Granted** status are treated as **Denied**.

Table 1. Permissions in Opportunity Detection

Permission	Description
View only	Access all of the user interface, in view-only mode.
Design triggers	Create workspaces and design trigger systems in batch mode. Allows the following. • Create, modify, and delete all trigger related resources. • Access Workspace, Component, Audience Level, Data Source, Named Value List pages. Can not access Server Groups and Deployment Configuration, and cannot set off a batch run.
Run for testing	• Deploy deployment configurations and run batch deployment configurations on server groups not designated for production. • Can access Server Group and Deployment Configuration pages,but can not designate a server group for production. Can not access any production related resources.
Run for production	• Deploy deployment configurations and run batch deployment configurations on any server group. • Perform all actions on the Server Groups, Deployment Configuration and Batch Run pages, including designating a server group for production.
Administer real time	Manage objects that the web service creates when Opportunity Detection is integrated with Interact to enable real time mode. Allows the following. • Delete workspaces and components created by the web service. • Start and stop real time deployment configurations and update their log level. The user with this permission alone can not start runs for real time deployment configurations. No one, even with this permission, can do any of the following. • Delete and update audience levels, data sources, named value lists, server groups, or deployment configurations created by the web service. • Create and deploy deployment configurations created by the web service.

Built-in roles in Opportunity Detection

Four built-in roles are included with Opportunity Detection.

In addition, you can create roles with permissions that you specify. See the *IBM Marketing Platform Administrator's Guide* for details on creating custom roles.

The following table shows the permissions assigned to each built-in role.

Table 2. Built-in roles in Opportunity Detection

Role	Permissions
OpDetectViewer	• View only
OpDetectTestDesigner	• View only • Design triggers • Run for testing
OpDetectProductionDesigner	• View only • Design triggers • Run for production
OpDetectAdmin	• View only • Design triggers • Run for testing • Run for production • Administer real time

Chapter 3. Workspaces in Opportunity Detection

To process data with a trigger system, you must create, deploy, and run a workspace. You create workspaces on the Workspaces page.

On the All Workspaces page, you can create or delete workspaces.

On the Component List tab of a workspace, you can add components and develop and manage the components to build a trigger system.

Creating a workspace

Use this procedure to create a workspace.

For information about running a workspace, see Chapter 11, "Deploying and running workspaces," on page 35.

1. Navigate to **Opportunity Detection > Workspaces** and click **Add**.

   The Add Workspace window opens, with three tabs: Properties, Component List, and Deployment & Batch Run.
2. Complete the fields on the Properties tab and click **Save**.
3. On the Component List tab, click **Add** to select the type of component you want to add to the workspace.

   When you select a component type, a component editor for the selected component type opens, where you can configure the component.

   See the remainder of this guide for detailed information about the various component types.
4. When you have finished creating a workspace, navigate back to the **Opportunity Detection > Workspaces** page, select the radio button for the workspace you just created, and click **Validate**.

   You cannot deploy and run a workspace until it passes the validation step. All components in the workspace must be completely configured for a workspace to be valid. A fully configured component has a green check mark in the component list.

Fields on the Workspace Component List tab

You add and manage components in a workspace on the workspace Component List tab.

Table 3. Fields on the workspace Component List tab

Field	Description
Add button	Click to select the type of component you want to add to the workspace. When you select a component type, a component editor for the selected component type opens, where you can configure the component. See the remainder of this guide for detailed information about the various component types.
Delete button	Click to permanently delete the selected component. A component that is being used by another component cannot be deleted.

Table 3. Fields on the workspace Component List tab (continued)

Field	Description
Filter	You can filter the component list by Description, Name, Name Space, Status, and type. Select a filter and then enter a search term and click **Filter**. Repeat to apply multiple filters. Click **Clear Filters** to remove all filters.
Status	Indicates whether a component is completely configured.
Name Space	Name Space for the component.
Name	Name of the component.
Type	Type of the component.
Origin	Indicates whether the component was created by a user or by the system. A component can be created by the system when Opportunity Detection is integrated with Interact.
Enabled	When Opportunity Detection is integrated with Interact for real time operation, you can enable or disable patterns. This field indicates the state of these pattern components. Components created in batch workspaces are always enabled.
Description	Description of the component, if the creator entered one.

Fields on the Workspace Properties tab

You complete the fields on the workspace Properties tab when you create a workspace. After the workspace is created, read-only fields provide information about the workspace.

Table 4. Fields on the workspace Properties tab

Field	Description
Name	Enter a name for the workspace
Description	Enter a description for the workspace. When Opportunity Detection is integrated with Interact for real time operation, a default, non-editable description appears in this field.
Partition	This field applies only when Opportunity Detection is integrated with Interact.
Origin	Indicates whether the workspace was created by a user or by the system. A workspace can be created by the system when Opportunity Detection is integrated with Interact. Read only.
Creation date	Date when the workspace was created. Read only.
Modified Date	Date when the workspace was most recently modified. Read only.
Status	Indicates whether the workspace has passed validation.

Chapter 4. Components in Opportunity Detection

Components are the basic building blocks for trigger systems in Opportunity Detection.

Broadly, there are three types of components: those that can send an event to other components, those that return a value that can be used by other components, and the Action component, which writes data to the Outcome table that can be used by external systems.

With these three types of components, you can build logic to suit your business requirements.

Event components

An event component is activated when it receives data that causes it to evaluate the data against its criteria.

An event component is said to 'fire' when it sends either a positive or negative event to activate a downstream event component. An event component fires a positive event when its criteria are met. Pattern components can also send a negative event when their criteria are not met.

The following event components have additional features.

- The Simple component is the only component that does not require an event to activate it. Instead, it processes all incoming data against its criteria. The Simple component requires at least on transaction data source. When it fires, it sends the first event in the series of events that a trigger system depends upon. This is why all trigger systems require a Simple component.
- The Action component is the only component that writes to a database when it fires.
- You can configure all three types of Pattern component to fire negative events rather than positive events. You can configure other event components to respond to negative events as well as positive events.

Data components

A Boolean Expression component is invoked by being referenced in another component, which then uses the value it returns in its own evaluation of the customer data.

Action components

When Action components fire, they write the data that you specify to the Outcome database table. These outcomes are available for use by external systems. All trigger systems require an Action component. If a workspace does not contain an Action component, it does not pass validation and you cannot deploy it.

Component type details and examples

These are the components that you can use to build trigger systems.

Event components

Table 5. Event components

Component Type	Description
Backward Inactivity	Listens for the occurrence of a specified event and then checks a specified prior time frame to see whether another specified event has occurred. The Backward Inactivity component fires if the specified prior event did not occur. **Example**: Fire when a customer makes use of the ATM after more than 1 month of not using any teller services.
Forward Inactivity	Listens for the occurrence of a specified event and then waits a specified time to see whether or not another specified event occurs. The Forward Inactivity component fires when the time period expires without the occurrence of the event it for which it was waiting. **Example**: Fire when a web-trade customer, who usually trades once a month, does not trade for two consecutive months.
Pattern (Match All, Counter, Weighted Counter)	Fires if specified events occur within a specified time frame. In addition, you can configure a Pattern component to fire a negative event if the pattern is no longer matched within the specified time frame. **Example**: Fire when customer uses his credit card four times a month during a three month period.
Simple	Fires if specified conditions based on transaction attributes are satisfied. A Simple is the only component type that is activated by incoming transactions and not by an event produced by another component. This is why every trigger system requires at least one Simple component as its starting point. **Example**: Fire when a customer makes a credit card purchase over $5000 or an international phone call to Italy.

Data components

Table 6. Data components

Component Type	Description
Boolean Expression	Evaluates data and returns True or False.

The Action component

The Action component is the only one that writes to the outcome table. When you configure an Action component, you define the data that is written to the outcome table when all the criteria specified in the components that the Action component depends upon are met.

Chapter 5. Basic attributes of components

You build expressions and configure incoming events and time spans when you work with several types of components. In addition, all component editors have a left panel where you configure or view basic attributes. Read this section to understand these commonly used component attributes.

Left panel of component editor

The left panel of a component editor contains the basic attributes of the component. The component type determines which of these attributes are present in the left panel.

Data Dependencies

A read-only list of data sources that are allowed for incoming events in this component and used in this component, and the audience level associated with these data sources. The list of data sources updates as you configure a component

The information in the Data Dependencies panel allows you to determine a component's access to transaction data sources.

When there is a mismatch between allowed data sources and the data sources used in the component, the component is not properly configured., and the trigger system where it is used does not pass validation.

Data source availability

Generally, transaction data sources used by upstream components are available for use in configuring a component, but there are additional considerations.

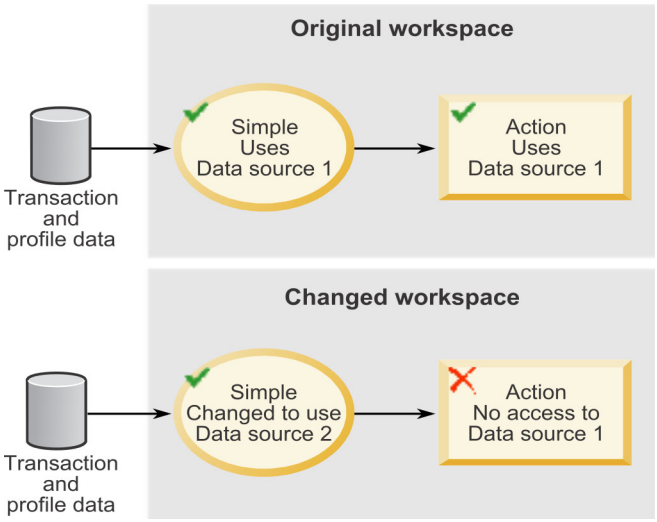
- If you choose a data source from among several transaction data sources in the Value Selector while you are configuring a component, you can access only that data source in subsequent uses of the Value Selector for that component.
- When a component fires and passes an event, it makes its transaction data available to any component it activates. All other transaction data sources are not available.
- Some components are potentially activated by events associated with more than one data source. In such cases, you cannot configure these components to use transaction data from an upstream component, because you cannot predict in advance which data source will be involved in activating that component.
- Because Forward Inactivity components fire only when the cancelling event does not occur before the time span completes, no downstream components have access to any transaction data sources.

If profile data exists for the audience level associated with the transaction data source, it is always available for use in configuring a component.

The following examples illustrate some of the cases when access to a transaction data source is restricted.

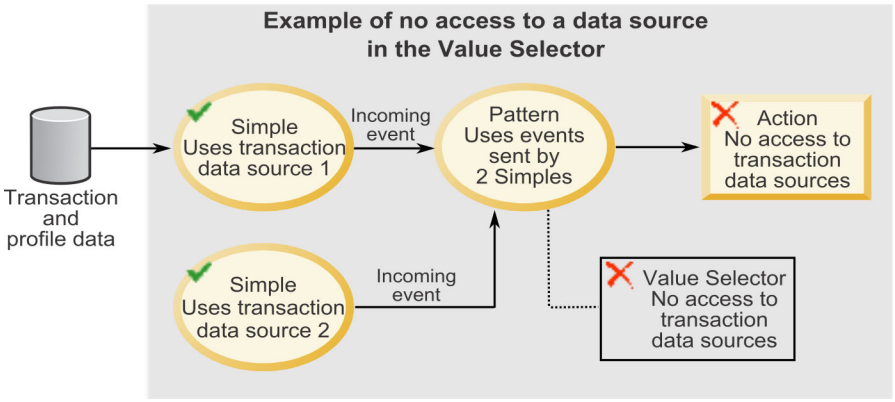
Example 1: Changing a data source

A change to one data source can affect a child data source. In this case, you would see in the Data Dependencies panel for the Action component that no data sources are available.



Example 2: Pattern components

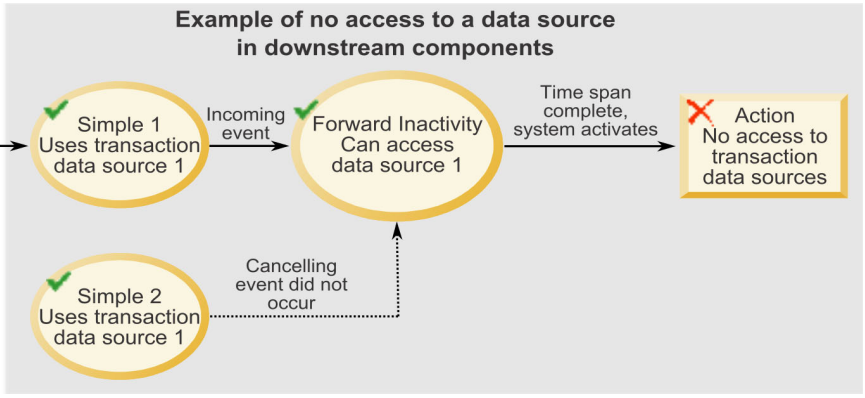
Pattern components such as Match All and Weighted Count have multiple incoming events. If two of the components sending the incoming events use different data sources, the pattern can not unambiguously determine its data source heritage. In the case illustrated in the diagram, the Pattern component's allowed data source would be set to None. If you open the Value Selector, you would not have access to any transaction data sources.



Example 3: Forward Inactivity components

A Forward Inactivity component uses incoming and cancelling events generated by two Simple components. In the Forward Inactivity's value selector, you can use the transaction data source made available through the component that supplies the incoming event. However, components that are downstream from the Forward Inactivity component have no access to this data source. This is because it is the system, and not an event generated by a component, that activates the Forward

Inactivity for firing when the time span elapses. At the moment when the Forward Inactivity component fires, the system has no access to any data source information.



Dependent Components

The name and type of any components that use this component. Use this information when you want to make a change to a component, to see what other components might be affected by the change. Read only.

Effective Window

Fields where you can set a start and end date during which an event component is active. Optional.

Firing Frequency

Fields where you set a limit on how often an event component produces its event. Optional.

The set of fields as a whole can be interpreted as follows.

Fire not more than *N* times in *N* time units, where a time unit is a combination of type and interval.

See “Time Span fields” on page 12 for additional details on time units.

Properties

Fields where you can enter a description of the component. Optional.

Incoming Event fields

You specify an incoming event to activate all components that send an event, except the Simple, which does not require an incoming event to activate it. The incoming event signals to the component that there is data for it to evaluate against its criteria.

Table 7. Incoming Event fields

Field	Description
Incoming Event	Select a positive or negative event, select the a component type, and select from the available components of that type. Finally, click **Add** to make your choice one of the incoming events for this component. If you use more than one incoming event, the component is activated when any one of the selected components fires and sends an event.

Time Span fields

You define time spans when you configure Forward Inactivity, Backward Inactivity, and all three types of Pattern components.

Table 8. Time Span fields

Field	Description
Quantity	You can enter a number directly or use the Value Selector to set the quantity of time units for this time span. Then use the time unit drop-down lists to complete the specification.
Select Value	Use the Value Selector to choose the quantity of time units for this time span. Then use the time unit drop-down lists to complete the specification.
Calendar, Rolling	This sets the type of the time span. The behavior of these time span types varies, depending on the component where they are used. See Chapter 10, “Backward Inactivity and Forward Inactivity components,” on page 29 and Chapter 9, “Pattern components,” on page 21 for details. Calendar time spans work in conjunction with time constants, which have defined meanings, as described in “Definition of calendar time units” on page 15.

Firing Condition fields

A firing condition is required for Simple components and is optional for Action components. You can define an expression to serve as a firing condition. The expression can be an existing expression component, or you can define an inline expression that is available only within the component where it is defined.

Table 9. Firing Condition fields

Field	Description
Type	Any Boolean Expression components used in this workspace are available for use in an expression. To use it in an expression, click **Add**.
The firing condition will be evaluated using	If you create more than one expression, they are evaluated using the logical operator you choose here. Options are **AND** and **OR**. You can also use AND, OR, and NOT logical operators within the inline expressions you create, to build nested logic.

Chapter 6. Building expressions using the Expression Builder

You always build expressions when you configure an expression component. You can optionally use expressions in Action and Simple components.

When you build an expression, you can compare strings and numbers, and perform simple math operations. The Expression Builder uses standard mathematical syntax. String constants must be surrounded by double quotes.

For operands, you can enter contstant values using the keyboard, or you can use the Value Selector to specify the following.

- Field values from a transaction data source.

  If you select a data source field, and if a Named Value list is configured for that field, you must use the Value Selector to add the value.
- Field values from a profile data source.
- Values calculated in expression components.

Your choices differ depending on the component you are working with, and on the data type you select.

In a Simple or Action component, you build expressions in the Firing Condition panel. In this panel, you must select either AND or OR as the logical operator used between the expressions you create in the Expression Builder. You do this using radio buttons labeled **The firing condition will be evaluated using**.

In an expression component, this top-level choice is absent. However, in all cases, you can use AND, OR, and NOT between expressions to create complex nested expressions.

Boolean, comparison, and math operators

You can use a variety of Boolean, comparison, and math operators to build an expression.

Boolean operators

Table 10. Boolean operators in the Expression Builder

Operator	Definition	Example
And	The component fires when all conditions are true.	Look for transactions that have a price per share greater than $99.99 AND were processed on June 15, 2001 AND were for accounts in New York.
Or	The component fires when any one of the conditions, any combination of the conditions, or all of the conditions are true.	Look for transactions that have a price per share greater than $99.99 OR were processed on June 1, 2001 OR were for accounts in New York.
Not	The component fires when all of the conditions are false.	Look for transactions that do not have a price per share greater than $99.99 AND were not processed on June 15, 2001 AND were not for accounts in New York.

Comparison operators

The following comparison operators are available. They are listed in their order of precedence.

- Equal
- Not Equal

  This is available only in the Simple Event component.
- Greater Than
- Greater Than or Equal To
- Less than
- Less Than or Equal To

Math operators

The following math operators are available. They are listed in their order of precedence.

- Multiply
- Divide
- Add
- Subtract
- Modulo (finds the remainder of division of one number by another)
- Exp (raises a number to the power of another number)

Order of operations

This is the order of precedence for operators.

- multiplication
- division
- addition
- subtraction

If you use a comparison in an expression that uses a Boolean operator, you must enclose the operand that follows the Boolean operator in parentheses whenever the result of the calculation would be ambiguous if you do not use parentheses. For example, you would need parentheses to clarify whether you want `trans.frequency < 3 AND trans.deposit > 2 * (prev.deposit + 50)` or `trans.frequency < 3 AND trans.deposit > (2 * prev.deposit) + 50`.

Value Selector fields

You specify values for expressions using the Value Selector.

Table 11. Value Selector fields

Field	Description
Data Type	Options vary depending on the component with which you are working. • Integer • Boolean • Double • String

Table 11. Value Selector fields (continued)

Field	Description
Type	Available options vary, depending on the selcted Data Type. • Constant • Data Source • Boolean Expression
Data Source	When you select Data Source as the Type, your configured data sources are available for selection. Availability depends on the component's data source dependency constraints.
Field	When you select Data Source as the Type, fields in the selected data source that match the selected data type are available for selection.
Value	When you select Constant as the Type, you can enter a value here.
Expression	The Boolean Expression components in your workspace are available for selection.

Definition of calendar time units

When you configure a calendar time span, you can select time units.

Calendar time units are defined as follows.

- A calendar week starts at midnight on Sunday.
- A calendar day starts at midnight.

You can specify end points for all time units, including irregular time units, such as months and years.

- Year: In addition to options that allow you to specify the second through 365th day of the year, you can also select **Last day of the year**, to account for leap years.
- Quarter: In addition to options that allow you to specify the second through 90th day of the quarter, you can also select the following options, because months have varying lengths.
  - **Last day of the quarter**
  - **Second to last day of the quarter**
  - **Third to last day of the quarter**
- Month: In addition to options that allow you to specify the second through 28th day of the month, you can also select the following options, because months have varying lengths.
  - **Last day of the month**
  - **Second to last day of the month**
  - **Third to last day of the month**

Expression examples

This section provides examples of using different operators to build expressions.

Example 1

This example shows a simple AND expression.

Look for:

transactions that have a price per share greater than $99.99

AND

are for accounts from New York

AND

were processed on June 15, 2013

Example 2

This example shows an OR expression with AND expressions.

Look for accounts that are from New York AND had transactions processed between December 25, 2013 and January 1, 2014

OR

Look for accounts that are from New Jersey AND that had transactions processed between December 25, 2013 and January 1, 2014

Example 3

This example shows an AND expression with OR expressions.

Look for:

accounts from New York OR from New Jersey OR from Connecticut that had transactions processed on June 15, 2014

AND

the price per share was greater than $99.99

Example 4

This example shows a NOT expression with OR expressions.

Note: Not(Boolean) is equivalent to Boolean = False. You can enter either one in the Expression Builder but, after you save the expression, the system always displays Not(Boolean).

NOT

where the price per share was less than $99.99

OR

that are associated with accounts that are not from New York

OR

that processed on June 15, 2014

Chapter 7. Simple Event Components

The sophisticated behavior patterns that IBM Opportunity Detection can track over time all have Simple components as their basic building block.

Simple is the only component type that is triggered by incoming transactions in the transaction data feed, and not by an event produced by another component. This is why every trigger system requires at least one Simple component.

Simple components watch for the occurrence of a single transaction that matches specified criteria. A Simple component fires a positive event when the criteria are met.

A Simple component must contain at least one comparison to a transaction data source field, in the form of a Boolean expression. This Boolean expression can be either an inline expression or a reference to an expression component. Without a transaction, the Simple component would never fire.

In its most basic form, the Simple Event component compares one field from a transaction data source to a specified value. For example, you could look at the value of the field *Transaction Code* found in the data source *Account Transactions* and test to see if it matches the string constant value *ACH Credit* .

Chapter 8. Action components

The purpose of an Action component is to send a notification that a specified behavior has occurred by writing data to a database, in the outcome table. An Action component fires and writes its outcome when it recognizes the occurrence of a single incoming event.

The audience ID is automatically included in the outcome table for each record that the Action component writes. In addition, you can include values created in the Value Selector.

Any of the following event-generating components can be the incoming event that triggers an Action component.

- Simple
- Pattern
- Forward Inactivity
- Backward Inactivity

You specify the outcome table by mapping the data source connector to the desired table.

Action component examples

Examples of the outcome an Action can write to the database table include the following.

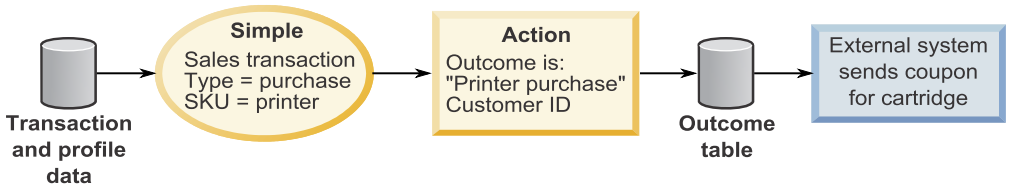
- Customer ID
- The identifier of the Action component that caused the outcome to be written
- The date/time stamp of when the Action component triggered
- A message consisting of a string you specify and optional additional information derived from the data available in the trigger system.

The following diagram illustrates the use of the Action component in a basic trigger system.

- A Simple component fires when a transaction data source contains a transaction of type "purchase" with an SKU that indicates that the customer purchased a printer.

  These criteria are specified in the Firing Condition panel using inline expressions.
- An Action component has the Simple component as its Incoming Event.
- The Simple component fires and sends an event that activates the Action component.
- The Action component fires and writes an outcome that consists of the string "Printer purchase" and the customer ID.

  The outcome is specified in the Outcome panel.



Outcome fields

In Action components, you can define the data that is written to the Outcome database table.

Table 12. Outcome fields

Field	Description
Message	You can enter any string, which will be written to the Outcome database table when the Action component fires.
Additional Information	You can use the Value Selector to include data from these sources to be included with the outcome message. • Profile and transaction data sources • Constants You can attach a label to each outcome value that you specify.

Chapter 9. Pattern components

You can use Pattern components to test whether one or more events for a given customer occur over a period of time. A Pattern component collects events that match its criteria and stores these events in the customer's state history record. When the specified pattern of events is met, the pattern drops those events from state history and fires a positive event.

Pattern components use a time span to set a boundary around the events it evaluates against its criteria. When you specify the time span, you can choose between Rolling and Calendar spans.

If events held in state history age out of this window, they are dropped and are no longer be evaluated against the pattern criteria.

Pattern component types

The following types of Pattern component are available.

- **Match All.** If all of the specified incoming events occur, the pattern fires a positive event.
- **Counter.** If the specified incoming events occur a specified number of times, the pattern fires a positive event.

  For example, you might look for cases where a customer makes 3 deposits.
- **Weighted Counter** You assign a score to each incoming event that you specify, and the pattern fires if a specified total score is reached.

  Unlike the Match All pattern, all of the specified incoming events do not have to occur; the total score is what determines whether the pattern criteria are met.

  For example, suppose you configure the pattern as follows.
  - You select incoming events and assign the following scores.
    - Incoming event 1 has a score of 1.
    - Incoming event 2 has a score of 2.
  - You specify a Weighted Counter value of 10.

  The event pattern would be true in any of the following cases.
  - Incoming event 1 occurs 10 times.
  - Incoming event 1 occurs 2 times and incoming event 2 occurs 4 times.
  - Incoming event 2 occurs 5 times.

Negative event modes in Pattern components

By default, a Pattern fires a positive event when its criteria are met, but it does not fire a negative event when the aging out of an event creates a negative condition. However, you can configure the behavior to include negative events.

When you enable negative events, the sequence of events that the component fires is always positive, negative, positive, negative, and so on.

You can select the mode for negative events: Immediate or Delay.

- In Immediate mode, the actions are as follows.

- When the Pattern component fires a positive event, it retains the events that fulfilled its criteria in the customer's state history. They continue to be used in evaluating the collection of events against the pattern criteria.
- The component fires a negative event as soon as the pattern is no longer fulfilled by collection of events being held for the customer in state history.
- The component continues to listen for incoming events and fires a positive event when the pattern is again fulfilled.

- In Delay mode, the actions are as follows.
  - When the Pattern component fires a positive event, it drops the events that fulfilled its criteria from the customer's state history.
  - The delay period, which is specified in the duration field, begins immediately. During the delay period, the component ignores incoming events.
  - At the end of the delay period, the component fires a negative event.

Calendar time span in Pattern components

You specify the calendar time span for the pattern using the **Time Span** section of the component editor.

The start and end points of the calendar time span are fixed. When the calendar time span is over, the span resets. The new start point is the next time unit following the previous end point.

The timestamp on the event, not the time when the event was processed, determines whether the event falls within the time span.

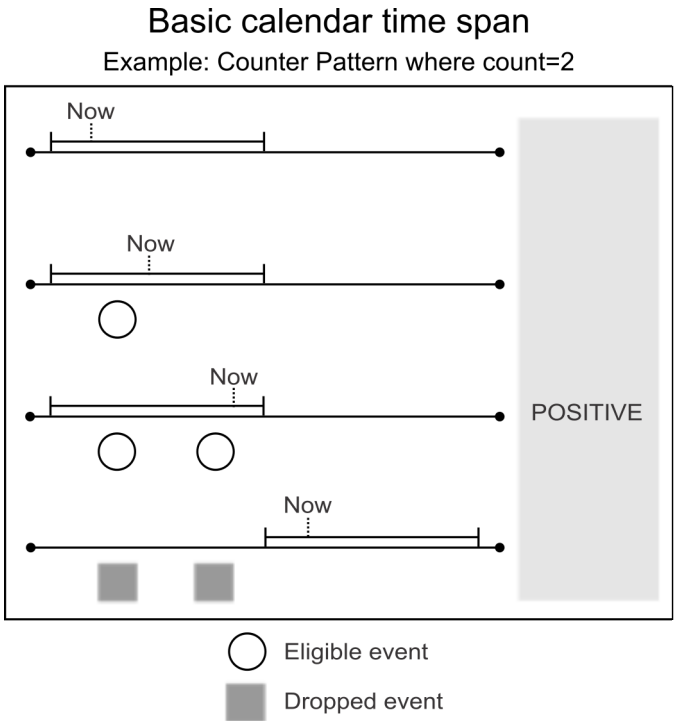
The calendar time span works in conjunction with time constants, which have defined meanings, as described in “Definition of calendar time units” on page 15.

For example, a week always begins on midnight on Sunday, and a day always begins at midnight. These definitions affect the behavior of calendar time spans, as shown in the following examples.

Example: Calendar time span, negative events not enabled

The following diagram illustrates how a basic calendar time span works when used in a Counter Pattern component where negative events are not enabled.

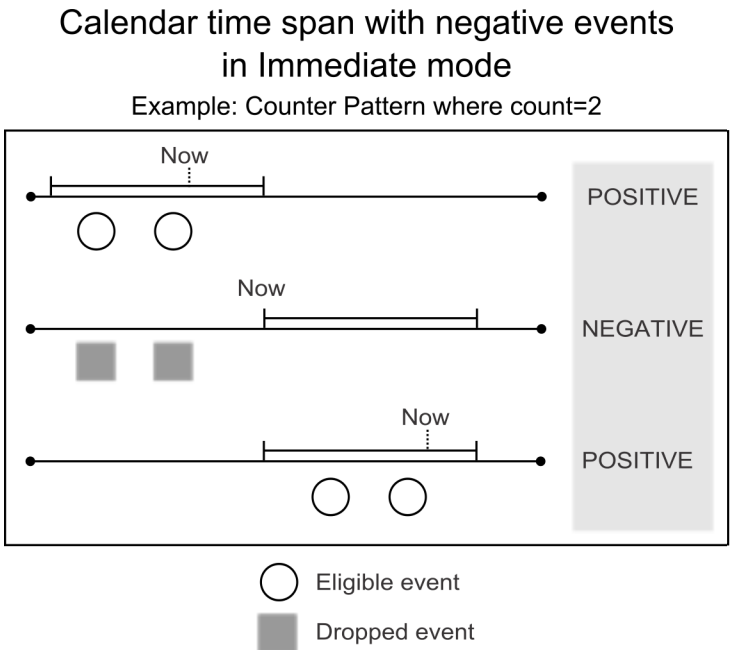
In the example, the component fires a positive event as soon as its criteria are met within the specified time span. Then it drops the events that caused the component to fire. When the time span expires, it resets.



Example: Calendar time span, negative events enabled in Immediate mode

The following diagram illustrates how a basic calendar time span works when used in a Counter Pattern component where negative events are enabled, and Immediate mode is selected.

In the example, as soon as the component's criteria are met, the component fires a positive event and drops the events that caused the component to fire. When the time span resets to the next calendar period, the component fires a negative event immediately.

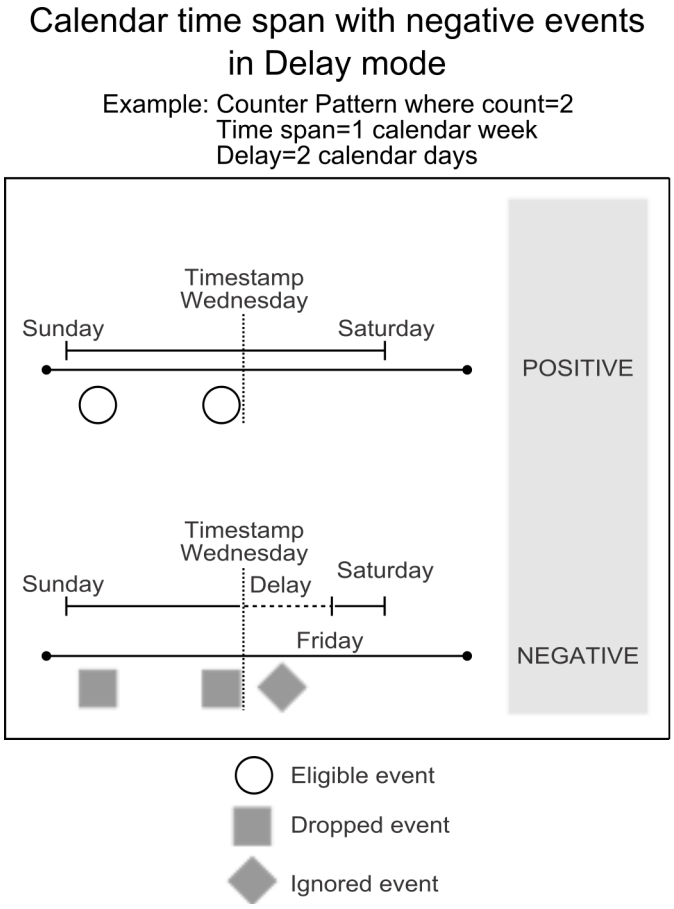


Example: Calendar time span, negative events enabled in Delay mode

The following diagram illustrates how a basic calendar time span works when used in a Counter Pattern component where negative events are enabled in Delay mode, and a time span of 1 calendar week and a Delay mode of 2 calendar days are selected.

In the example, the actions are as follows.

- The event that completes the component's pattern has a timestamp of Wednesday. The component fires a positive event immediately and drops all events from the customer's state history.
- The 2 day delay period begins. During the delay period, eligible incoming events are ignored.
- At the end of the delay period, on Friday, the component fires a negative event and begins to collect events again.
- The time span resets at midnight on Sunday (not shown in the diagram).



Rolling time span in Pattern components

You specify the rolling time span for the pattern using the **Time Span** section of the component editor.

Events must occur within the specified time span to be eligible for evaluation against the pattern criteria. The start of the time span moves forward in time, but the span you specify always determines the length of the window.

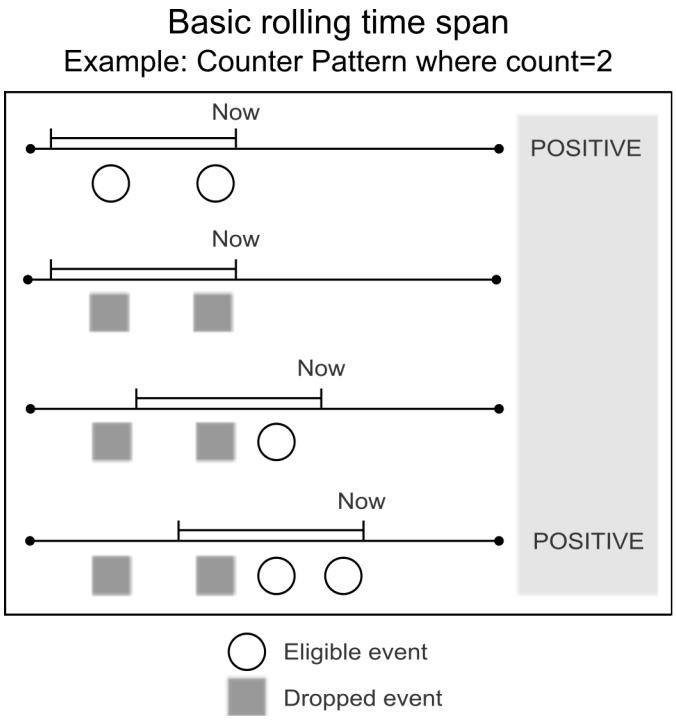
In contrast to the calendar time span, with the rolling time span, events are retained after a positive event is fired. They are dropped only when they age out of the time span.

The timestamp on the event, not the time when the event was processed, determines whether the event falls within the time span.

Example: Rolling time span, negative events not enabled

The following diagram illustrates how a basic rolling time span works when used in a Counter Pattern component where negative events are not enabled.

In the example, as soon as the component's criteria are met, the component fires a positive event. It drops the events that caused the component to fire. As the time span rolls forward, it drops the event that ages out and continues to collect eligible events.



Example: Rolling time span, negative events enabled in Immediate mode

In Immediate mode with the rolling time span, the pattern must fire its first positive event before it ever fires a negative event.

The following diagram illustrates how a basic rolling time span works when used in a Counter Pattern component where negative events are enabled, and Immediate mode is selected.

In the example, after the first positive event is fired, the component retains the events until they age out. It fires a negative event as soon as an event ages out so that the pattern is no longer fulfilled. The pattern continues to listen for incoming events and fires a positive event when the pattern is again fulfilled.

Rolling time span with negative events in Immediate mode

Example: Counter Pattern where count=2

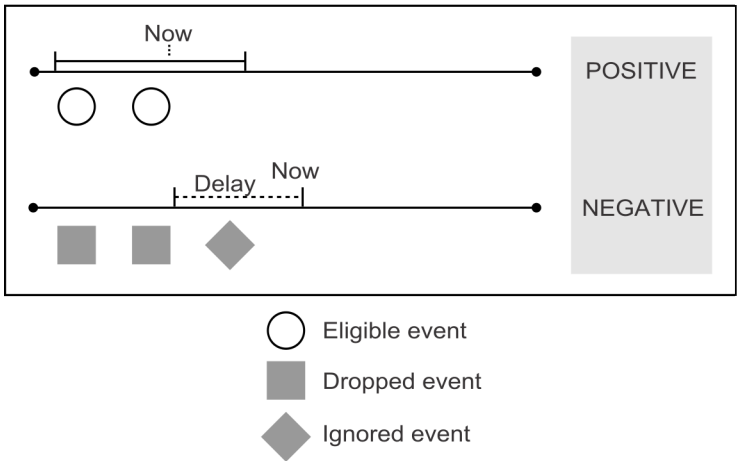
Example: Rolling time span, negative events enabled in Delay mode

In Delay mode with the rolling time span, the pattern must fire its first positive event before it ever fires a negative event.

The following diagram illustrates how a basic rolling time span works when used in a Counter Pattern component where negative events are enabled, and Delay mode is selected.

In the example, after the first positive event is fired, the component drops its events, the delay period begins, and the component ignores incoming events. It does not fire the negative event until the delay period you specify is over. At that point the component begins to listen for incoming events again.

Rolling time span with negative events
in Delay mode
Example: Counter Pattern where count=2



Pattern Behavior fields

Pattern behavior options allow you to enable the pattern to fire a negative event (by default patterns fire only positive events). They also allow you to delay firing the negative event.

Table 13. Pattern Behavior fields

Field	Description
Incoming events must occur this many times	This field is available only on the Counter Pattern component. Enter an integer to specify how many times the incoming event must occur before the pattern fires a positive event.
Enable negative events	By default, pattern components send only positive events. This checkbox enables the pattern to send negative events.
Delay Mode and Immediate Mode	Delay mode allows you to delay the firing of the negative event for a specified period of time. For details, see “Negative event modes in Pattern components” on page 21.

Chapter 10. Backward Inactivity and Forward Inactivity components

You can use a Backward Inactivity component when you want to verify that some event did not occur within a specified period of time before an incoming event. You can use a Forward Inactivity component to set a timer when it receives an incoming event, and optionally specify another event that turns off the timer.

Backward Inactivity components

When a customer withdraws $10,000 from her account, you might want to determine whether she has made any other withdrawals larger than $5,000 within the past two months. For this, you would use a Backward Inactivity component.

The Backward Inactivity component requires the following settings.

- Blocking event: the event that activates the component and sets the start of the time span. The timestamp is saved in the customer's state history.

  Alternatively, you can check **Use the firing of this BI as a Blocking Event**. In that case, the date of the component firing is used as the Blocking Event date.
- Time span: the period of time before the blocking event that is used as a boundary when the system compares the time stamp of the triggering event to the time stamp of the blocking event.
- Triggering event: the event whose timestamp is compared with the blocking event's timestamp to determine whether the time span between the two events is greater than the time span specified in the component. If the date of the triggering event is within the backward-looking time span, the component does not fire.

You can set additional specifications to modify the way the component behaves after it has been activated by the blocking event.

Additional options

In addition, you can use either or both of the following options.

- **Use the firing of this component as a blocking event**: Usually the blocking event and triggering event are different. But if you select **Use the firing of this BI as a Blocking Event**, then the timestamp of the component firing is used in place of the timestamp of the blocking event.
- **Use creation date if no Blocking Event**: Usually the blocking event occurs before the triggering event, but it might occur after the triggering event. By default, if the triggering event occurs, and no blocking event occurred before the triggering event, the component fires. If you check **Use creation date if no Blocking Event**, the component uses its creation date as the start of the time span if the blocking event did not occur.

Examples of Backward Inactivity Components

This section provides examples of Backward Inactivity components.

Example 1: Store purchase after no online purchases

When a customer makes a store purchase, this component evaluates the data for that customer to see if he made an online purchase in the previous 6 months. If an online purchase has not occurred in the past six months, the component fires for that customer.

This component has the following parameters.

- Triggering event—customer makes a store purchase
- Blocking event—online purchase
- Time span—prior 6 months

Example 2: First major deposit after a long interval

When a customer makes a major deposit, this component evaluates the data for that customer to see if she made any other deposits or withdrawals in the past month. If these activities have not occurred in the past month, the component fires for that customer.

This component has the following parameters.

- Triggering event—make a major deposit
- Blocking event—major deposit
- Time span—1 month

Forward Inactivity components

When a customer makes a purchase, you might want to use a Forward Inactivity component to determine whether he returns within the next month.

The Forward Inactivity component requires the following settings.

- Incoming Event: the event that activates the component and sets the start of the time span. The date stamp in the transaction that is being processed for the incoming event is used as the starting point of the time span.
- Time Span: the period of time after the date stamp in the incoming event before the component fires its positive event.

You can set additional specifications to modify the way the component behaves after it has been activated by the incoming event.

The reset option

You can add an optional reset to the component by checking the **Reset the timer each time the incoming event occurs** box in the Incoming Event panel. It changes the behavior of the Forward Inactivity component as follows.

- Reset option is not enabled: After the component is activated by an incoming event, the component ignores any additional instances of the incoming event that have a date stamp that falls within the time span, until the time span is over.
- Reset option is enabled: After the component is activated by an incoming event, if the component receives an additional instance of the incoming event that has a date stamp that falls within the time span, it resets the time span.

Cancelling event

You can set an optional cancelling event. If a cancelling event occurs after the component is activated, it stops the clock on the time span and the component again can respond to an incoming event. The date stamp in the transaction that is being processed to produce the canceling event is what the system uses for evaluation against the time span.

Time span, reset, and other options in Forward Inactivity components

You can set either a calendar or rolling time span in Forward Inactivity components.

For both calendar and rolling time spans, the date stamp in the transaction that is being processed for the incoming event is used as the starting point of the time span.

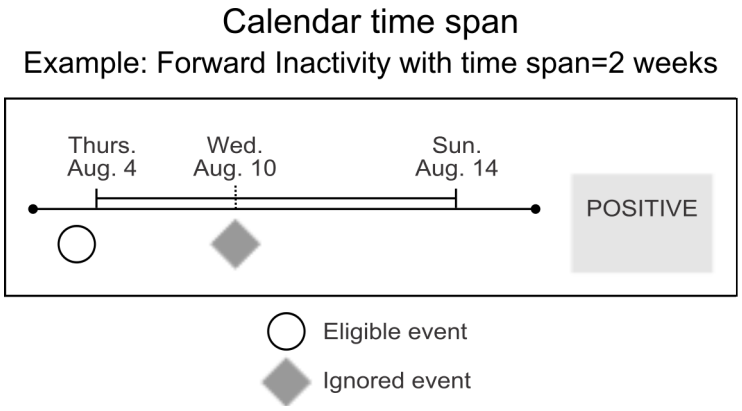
Calendar time span

The calendar time spans have defined meanings, as described in "Definition of calendar time units" on page 15.

In the example diagram, a Forward Inactivity has a time span of two calendar weeks, and the component is activated by a transaction with a date stamp of Thursday, August 4. No cancelling event is specified.

The first week of this calendar time span ends at midnight on Sunday, August 7, only three days later. When the second week ends at midnight on Sunday, August 14, the time span expires and the component fires. This is because a week as defined by the time constant always begins at midnight on Sunday.

The component in the example ignores any additional incoming events because reset is not enabled.

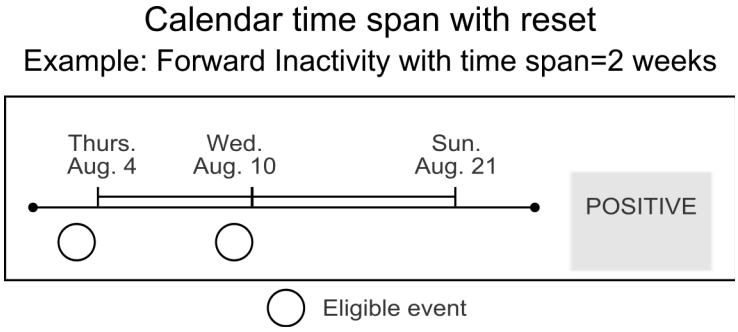


Calendar time span with reset

If reset is enabled, the component continues to listen for incoming events even after it is activated.

In the example diagram, a Forward Inactivity has a calendar time span of two weeks, and the component is activated by a transaction with a date stamp of Thursday, August 4. No cancelling event is specified.

With reset enabled, the component would reset its time span if a second incoming event arrives with a transaction date that falls within the time span. Suppose this transaction date is Wednesday, August 10. The time span resets, and the first week ends at midnight on Sunday, August 14, only four days later. When the second week ends at midnight on Sunday, August 21m the time span expires and the component fires.



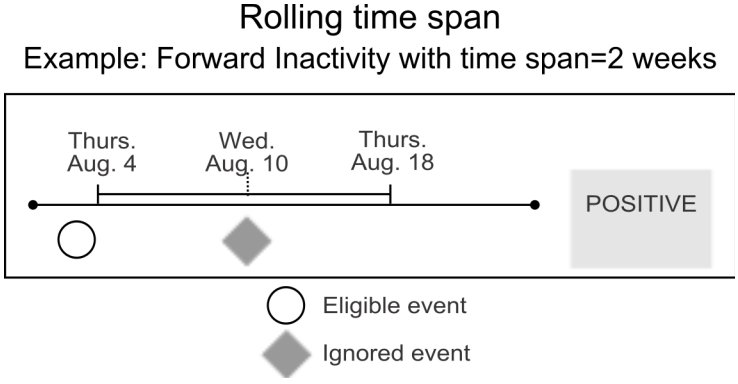
Rolling time span

The rolling time span works differently in Forward Inactivity components from the way it works in Pattern components. The length of the span remains constant, but the start date does not move forward in time as it does with a Pattern.

In the example diagram, a Forward Inactivity has a rolling time span of two weeks, and the component is activated by a transaction with a date stamp of Thursday, August 4. No cancelling event is specified.

The time span expires and the component fires 14 days later, on Thursday, August 18.

The component in the following diagram ignores any additional incoming events because reset is not enabled.

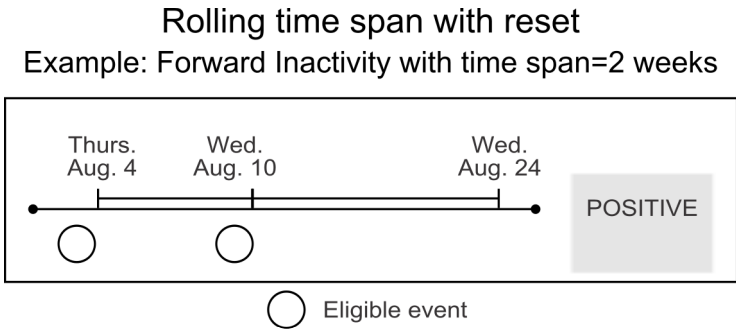


Rolling time span with reset

If reset is enabled, the component continues to listen for incoming events even after it is activated.

In the example diagram, a Forward Inactivity has a rolling time span of two weeks, and the component is activated by a transaction with a date stamp of Thursday, August 4. No cancelling event is specified.

With reset enabled, the component would reset its time span if a second incoming event arrives with a transaction date that falls within the time span. Suppose this transaction date is Wednesday, August 10 at 9:47AM. The time span resets, and the first week of this reset time span ends at at 9:47AM on Wednesday, August 17. The time span expires and the component fires on at 9:47AM on August 24.



Additional time settings

You can use the settings in the Effective Window and Firing Frequency panels to refine the component's behavior. See “Effective Window” on page 11 and “Firing Frequency” on page 11 for details.

Examples of Forward Inactivity components

This section provides examples of Forward Inactivity components.

Example 1: No additional deposits in 1 month

On June 10, a customer makes a complaint through a bank's call center.The Forward Inactivity component is activated and starts monitoring to detemine whether that customer makes any deposits. If the customer makes a deposit, the Forward Inactivity does not fire. If the customer does not make any deposits, the component fires for that customer so a follow-up can be initiated.

This component is configured as follows.

- Incoming event—customer makes a complaint
- Cancelling Event—customer makes a deposit
- Time Span—one rolling month

Example 2: No additional deposits this month

On June 10, a customer makes a deposit. The Forward Inactivity component starts monitoring to determine whether that customer makes another deposit. If another deposit does not occur by July 1, the component fires for that customer.

This component is configured as follows.

- Incoming Event—make a deposit
- Cancelling Event—makes a deposit
- Time Span—one calendar month

Chapter 11. Deploying and running workspaces

On the Deployment & Batch Run tab of a workspace, you create deployment configurations for workspaces, deploy the configurations, and run workspaces. You can select which server group to use, change default database connection mappings, manage run parameters, and view deployment and batch run history.

Before a workspace can process data, you must deploy it.

When you deploy a workspace, you create a deployment configuration and send the workspace, with all of its component configurations, to the Streams server, where it is compiled into a Streams application.

The deployment configuration allows you to select a server group and to change default data source mappings. Your changed mappings apply only within the deployment configuration and do not affect the global, default mappings for the server group.

When a workspace is successfully deployed on the Streams server, you can select a saved deployment configuration to run it.

Deployment configurations

Deployment configurations allow you to select the Streams server where the workspace processes data. You can also change the default data source mappings.

Opportunity Detection environments typically include at least one test server group and one production server group. For any workspace, you can create a deployment configuration for each server group configured in your system.

Using different deployment configurations, you can test components against new data sources, test new components, test edits to existing components, and build new components without affecting the production environment.

For example, you can create a deployment configuration that allows you to deploy a workspace to the test server group to test it, and you can create another configuration that allows you to deploy the workspace to the production server group for production runs. By changing data source mappings, you can separate the test data from the production data in your state history and outcome tables.

For best performance, you would normally run only a single workspace on a production server group.

Data source mapping

Default mappings of data sources and database tables are set when server groups are configured. When you create a deployment configuration, you can change these mappings.

These changes apply only within the deployment configuration; they do not affect the default mappings for the server group.

For example, you might want to use one set of State and Outcome tables with the Test server group in one workspace, and another set of State and Outcome tables with the Test server group in another workspace. You can do this by changing the connector or the database connection on the Data Source Mapping tab of the Details panel for the deployment configuration.

Server group availability

The server groups available for selection in a deployment configuration are those that are not yet mapped in any deployment configuration for the workspace you are working with. You can use a server group only once for each workspace, but you can use the same server group in deployment configurations for different workspaces.

Required permissions

The permissions assigned to a user determine which server groups they can access when creating a deployment configuration, as follows.

- A user with the **Run for testing** permission sees only server groups designated for test usage. The built-in **OpDetectTestDesigner** role has this permission.
- A user with the **Run for production** permission sees only server groups designated for production usage. The built-in **OpDetectProductionDesigner** role has this permission.
- To have access to both test and production server groups, a user must have both of the permissions listed above. The built-in **OpDetectAdmin** role has both of these permissions.

In addition, your system administrator may also create custom roles that include the required permissions.

Creating a deployment configuration

Use this procedure to create a deployment configuration.

For information about running a workspace, see Chapter 11, "Deploying and running workspaces," on page 35.

1. Navigate to the **Deployment & Batch Run tab** of the workspace for which you want to create a deployment configuration and click **Add**.

   The Details panel opens, with three tabs: Properties, Data Source Mapping, and History.
2. Complete the fields on the Properties tab, select a server group, and click **Save**.
3. If you want to override the default data source mapping for the server group, do the following.
   a. On the Data Source Mapping tab, select the data source you want to change
   b. Clear the Server Group Default checkbox.
   c. Select an alternate mapping for the data source and click **OK**.

   These changes apply only within the deployment configuration; they do not affect the default mappings for the server group.

Running a workspace

Use this procedure to run a workspace.

1. Verify that the workspace is valid by navigating to **Opportunity Detection > Workspaces** and checking its status on the All Workspaces page.

   You cannot deploy and run a workspace until it passes the validation step. All components in the workspace must be completely configured for a workspace to be valid.

   If the workspace is not valid, correct any problems, and then on the All Workspaces page, select the radio button for the workspace you want to run and click **Validate**
2. Click the name of the workspace you want to run to open the workspace page.
3. On the Batch Run panel, select the deployment configuration you want to run.
4. Complete run parameters as desired.
5. Click **Start**.

Fields on the Deployment Configuration & Batch Run page

You create deployment configurations and run workspaces from the Deployment & Batch Run tab.

Table 14. Fields in Deployment Configuration

Field	Description
Deployment Configuration	
Add button	Click to create a deployment configuration in the Details panel. See Details tabs, below, for information on the fields you use to create a deployment configuration.
Deploy button	Click to send the workspace to the Streams server, where it is compiled into a Streams application. When a workspace is successfully deployed on the Streams server, it is ready to be started.
Stop and Start buttons	You can stop the deployment process and re-start it by clicking the Stop and Start buttons.
Delete button	Click to delete the selected deployment configuration.
Status	Indicates whether all data sources are mapped for the deployment configuration. You can not deploy an incomplete deployment configuration.
Deployment Configuration	Name of the selected deployment configuration.
Server Group	Name of the server group the selected deployment uses.
Deployed	Indicates whether the selected configuration has been deployed.
Version	For batch runs, this number shows the version number of the most recent successful deployment. For real time mode, the number shows the version most recently deployed, re-deployed, or undeployed from IBM Interact.
Workspace modified on	Date and time when the workspace was last modified. By comparing this date and time to the date and time in the **Deployed On** column, you can determine whether the latest version of the workspace is deployed.
Deployed On	Date and time of the most recent successful deployment of the selected deployment configuration.
Details: Properties tab	
Name	Enter a name for the deployment configuration.

Table 14. Fields in Deployment Configuration (continued)

Field	Description
Server Group	Select from a list of available server groups.
Usage	The usage entered in the definition of the selected server group is shown here.
System Log Level	Select a system log level. Options are as follows, listed in ascending order of detail. • Fatal • Error • Warning • Information • Trace • Debug Raising the log level can affect performance. You can override the log level during a run by changing the level in this field and clicking **Update log level**.
Script file name (Optional)	If you use a notification batch file, enter the file name here.
Details: Data Source Mapping tab	
Status	Indicates whether a data source is mapped to a connector.
Data Source	Data sources used in the workspace are listed here. Click a link to view the File or Table Data Source Mapping dialog box. You can retain the default mapping used in the server group selected on the Properties tab, or you can deselect the **Server Group Default** check box to change the mapping of data source with connector.
Type	Lists the configured data source type: Transaction, State, or Outcome.
Connector	Lists the connector used with the data source for this deployment configuration.
Connector Type	Lists the connector type of the data source: Table, File, or TCP. TCP is used only when Opportunity Detection is integrated with Interact. See "Batch and real time modes" on page 1 for details.
Database	Lists the database configured for the data source. Applies only to Outcome and State data sources.
Default	Indicates whether the data source uses the default mappings for the server group selected on the Properties tab.
Details: History tab	
From Date, To Date, Get History button	Select a date range and click **Get History** to see details for the successful deployments of the selected deployment configuration.
Status	For batch mode, indicates whether the version of the selected deployment configuration was successfully deployed. For real time mode, Undeploy and Redeploy are additional actions for which success or failure can be indicated.
Version	The number of the deployment for which the row provides details. Includes successful and failed deployments.

Table 14. Fields in Deployment Configuration (continued)

Field	Description
Action	For stand-alone Opportunity Detection, the only action is Deploy. When Opportunity Detection is integrated with Interact, Undeploy and Redeploy are additional actions.
Date	Date and time when the listed action was completed.
Message	Additional details for the listed action.

Table 15. Fields in Batch Run

Field	Description
Batch Run	
Deployment Configuration	Select a deployment configuration to run and click **Start**.
Run Parameters	
Feed Path	Location of the feed files.
Artificial Transaction	Check this box if your workspace uses any Forward Inactivity components. The system uses the timestamp of the artificial transaction to determine if the time span of any Forward Inactivity component has elapsed. If the time span has elapsed, the component fires. You can select the following options. • Off - No artificial transaction is set. • End of Day - Sends an artificial transaction to indicate the end of day at the end of each unique transaction date in the feed files. • End of Run - Sends an artificial transaction after all transactions are sent for a user, regardless of how many transaction dates are in the feed files.
System Log Level	You can change the log level set for the deployment configuration by selecting a log level in this field and then clicking **Update Log Level** in the Details panel. This setting is temporary unless you click **Save** in the Details panel.
Run in Recovery	If a run fails, and you re-start the run after fixing any problems, the system attempts to pick up the run where it left off if you select this box.
Batch Notification	Select this box if you have a batch notification file and you want to receive notifications.
Most Recent	
Stop Run button	Click the **Stop Run** button to stop the most recent run.
Run id	Unique identifier for the most recent run.
Detail id	A counter that increments each time a recovery run occurs.
Deployment Configuration	The deployment configuration used for the run.
Start Time	Date and time of the beginning of the run.
End Time	Date and time of the end of the run.
Run Time	Duration of the run.

Table 15. Fields in Batch Run (continued)

Field	Description
State	During the run, this column may display a variety of interim states. When a run ends, the state is one of the following. • Run Success • Run Success with Errors • Run Failed
Run History	
Deployment Configuration	Select a deployment configuration for which you want to retrieve run history.
From Date, To Date, Get Run History button	Enter or select a date range and then click **Get Run History** to retrieve information about past runs.
Result	The final status of the run. • Run Success • Run Success with Errors • Run Failed
Run Id	Unique identifier for the run.
Detail Id	A counter that increments each time a recovery run occurs.
Run Type	Indicates whether the run is a normal or recovery run.
Start Time	Date and time of the beginning of the run.
End Time	Date and time of the end of the run.
Run Time	The duration of the run.

About batch files

The Opportunity Detection engine can run a batch file when it completes a run. You can issue different notifications for each successful and failed engine run.

An example file is shipped with the product; you must edit this file as needed to perform the appropriate actions for your system when theOpportunity Detection engine completes a run. If the batch file is empty, no further processing takes place.

The batch file must be a shell script with the `.sh` extension.

When you run the Opportunity Detection engine from the Deployment & Batch Run tab in a workspace, check the **Batch Notification** checkbox and specify the file name to run a batch file when the process completes.

Batch file usage examples

- After an engine process runs successfully, the notification file could call an external process to act on the reported outcomes.
- After an engine process fails to complete, the notification file could send a notification of the failure to the appropriate people.

Batch file required location

The batch file that Opportunity Detection invokes at the end of engine runs must be located on the machine where Streams is installed, in the `scripts` directory under your Opportunity Detection installation.

Before you contact IBM technical support

If you encounter a problem that you cannot resolve by consulting the documentation, your company's designated support contact can log a call with IBM technical support. Use these guidelines to ensure that your problem is resolved efficiently and successfully.

If you are not a designated support contact at your company, contact your IBM administrator for information.

Information to gather

Before you contact IBM technical support, gather the following information:

- A brief description of the nature of your issue.
- Detailed error messages that you see when the issue occurs.
- Detailed steps to reproduce the issue.
- Related log files, session files, configuration files, and data files.
- Information about your product and system environment, which you can obtain as described in "System information."

System information

When you call IBM technical support, you might be asked to provide information about your environment.

If your problem does not prevent you from logging in, much of this information is available on the About page, which provides information about your installed IBM applications.

You can access the About page by selecting **Help > About**. If the About page is not accessible, check for a `version.txt` file that is located under the installation directory for your application.

Contact information for IBM technical support

For ways to contact IBM technical support, see the IBM Product Technical Support website: (http://www.ibm.com/support/entry/portal/open_service_request).

Note: To enter a support request, you must log in with an IBM account. This account must be linked to your IBM customer number. To learn more about associating your account with your IBM customer number, see **Support Resources > Entitled Software Support** on the Support Portal.

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 about 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 Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (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 websites 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 Corporation
170 Tracer Lane
Waltham, MA 02451
U.S.A.

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.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

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.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

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.

Privacy Policy and Terms of Use Considerations

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. A cookie is a piece of data that a web site can send to your browser, which may then be stored on your computer as a tag that identifies your computer. In many cases, no personal information is collected by these cookies. If a Software Offering you are using enables you to collect personal information through cookies and similar technologies, we inform you about the specifics below.

Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user’s user name, and other personal information for purposes of session management, enhanced user usability, or other usage tracking or functional purposes. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

Various jurisdictions regulate the collection of personal information through cookies and similar technologies. If the configurations deployed for this Software Offering provide you as customer the ability to collect personal information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for providing notice and consent where appropriate.

IBM requires that Clients (1) provide a clear and conspicuous link to Customer’s website terms of use (e.g. privacy policy) which includes a link to IBM’s and Client’s data collection and use practices, (2) notify that cookies and clear gifs/web beacons are being placed on the visitor’s computer by IBM on the Client’s behalf along with an explanation of the purpose of such technology, and (3) to the extent required by law, obtain consent from website visitors prior to the placement of cookies and clear gifs/web beacons placed by Client or IBM on Client’s behalf on website visitor’s devices

For more information about the use of various technologies, including cookies, for these purposes, See IBM’s Online Privacy Statement at: http://www.ibm.com/privacy/details/us/en section entitled “Cookies, Web Beacons and Other Technologies."

Printed in USA