IBM Unica Detect
Version 8 Release 5
October 28, 2011

User's Guide

Note
Before using this information and the product it supports, read the information in "Notices" on page 107.

This edition applies to version 8, release 5, modification 0 of IBM Unica Detect (product number 5725-D16) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Chapter 1. Contacting IBM Unica 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® Unica technical support. Use the information in this section to ensure that your problem is resolved efficiently and successfully.

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

Information you should gather

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

- A brief description of the nature of your issue.
- Detailed error messages 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" below.

System information

When you call IBM Unica 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 Unica applications.

You can access the About page by selecting **Help > About**. If the About page is not accessible, you can obtain the version number of any IBM Unica application by viewing the `version.txt` file located under each application's installation directory.

Contact information for IBM Unica technical support

For ways to contact IBM Unica technical support, see the IBM Unica Product Technical Support website: (http://www.unica.com/about/product-technical-support.htm).

Chapter 2. About IBM Unica Detect

IBM Unica Detect enables you to look for specific customer behaviors and patterns represented in your customer data and respond to them. You define the transactions and patterns that Detect looks for, and the actions that result when those criteria are met.

Detect can recognize complex patterns of activity discovered in incoming streams of transactions. The strengths of Detect allow your business to:

- make decisions on incoming data extremely quickly
- configure the system to receive any kind of transactional input and to access existing static data
- easily create and modify the central rules that control the triggering behavior using an intuitive web-based interfaces

About trigger systems

A trigger system monitors incoming data, captures it, and if certain conditions are met, fires an outcome action. The main building blocks used to define those events are called components.

Each trigger system defines how to pick out important events within your transaction and static data. It defines the patterns that you want to capture, and it defines how to package up gathered information and attach it to the triggered outcome.

Building trigger systems

You can build trigger systems in two ways.

- You can define your own trigger systems in the Workspaces area of Detect using either the graph view or list view.
- You can use trigger system libraries defined by IBM Unica . You can import the trigger systems in these libraries and use them as delivered from IBM Unica , or you can customize them.

What every trigger system needs

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

- A Simple Event component
- A data source
- A container or pattern component
- An action component

You can combine these components within a workspace to build a trigger system.

Data sources are described in the *IBM Unica Detect Administration Guide* .

Chapter 3. Introduction to Building Trigger Systems

A trigger system in IBM Unica Detect monitors incoming data, captures it, and if certain conditions are met, fires an action component. The building blocks used to create trigger systems are called components. You create components and combine them to create trigger systems in workspaces. Each workspace can include one or more trigger systems.

Detect is capable of combining many trigger systems to recognize sophisticated patterns and produce a large set of outcomes based on the occurrence of those patterns. To help you manage the complexity of these systems, Detect provides the runset feature.

About using runsets to manage trigger systems

A runset is Detect's mechanism for assembling a set of triggers for execution in your production environment. Runsets can assemble trigger systems from multiple workspaces.

A good approach to building and using trigger systems is to create a series of workspaces that each contain only one trigger system with one outcome action. After you have built and tested your individual trigger systems, you can then use runsets to run them together them in various ways.

When you use the Detect Engine to run a published runset, Detect executes all the trigger systems referenced by the runset. Detect creates a unique outcome table associated with each runset.

About testing and running trigger systems

To test a workspace or runset, you run it in the Engine Manager area of the user interface. To run a workspace or runset in production, you should first publish it, and then run the Detect Engine from the command line. For details about publishing and running trigger systems, see Chapter 16, "Publishing and Runing Trigger Systems," on page 97.

About workspaces

You can create multiple workspaces where you can create and test collections of components. You can test components against new data sources, test new components, test edits to existing components, and build new components without interfering with the live, running environment.

After testing is complete, you can do either of the following.

- You can use a runset to reference trigger systems in one or more workspaces, publish the runset, and then perform production runs.
- You can publish any workspace and then perform production runs.

See the *IBM Unica Detect Administrator's Guide* for information on performing test and production runs.

You can perform the following tasks in the Detect Workspaces area.

- **Manage the list of workspaces**—You can add a new workspace, or copy or delete an existing workspace. You can also rename and describe the workspaces.
- **View and manage the components within a workspace**—By selecting an individual workspace, you access its Workspace Editor. From there you can develop and manage the components to build a trigger system within the workspace.
- **Publish a workspace**—You can promote a workspace to production and archive the last active workspace. Only one workspace at a time can be published.

Two ways to create and edit components

You create new components and modify existing ones in the Component Editor. Detect gives you the following two options for accessing the Component Editor.

- On the Component Graph tab of a workspace, right-click the component on the graph and select Edit in the context menu.
- On the Component List tab of a workspace, click the name of the component.

About the Component Graph

In every workspace, you can view and work with trigger systems in graph form. The graph clearly indicates the event flow and the relationships among the components. The graph provides one of the two ways you can access the Component Editor to build trigger systems.

Detect automatically generates a graph of the workspace. The generated graph hides clutter and lays out all of the components and component relationships in the trigger systems.

You can optionally rearrange and simplify the graph and then save the modified graph as a custom view. Detect updates the generated view and all custom views when you make changes to trigger systems in a workspace.

About custom views

The generated graph display is recreated each time you view the Component graph tab. To keep any changes you make to the way this graph is displayed, save your modified graph as a custom view.

You can make as many views as you want, and set any of them to be the default view. You can use these views to focus on different aspects of your trigger systems as an aid to your own understanding, and to communicate with others.

To set a custom view as the default view

Do one of the following.

- On the Component Graph tab of the Workspace Editor, with the custom view displayed, click the **Change View Settings** icon  select the **Default View** checkbox.
- On the Properties tab of the Workspace Editor, click **Views**, select the desired custom view as the default view, and click **Save Default View**.

About using Pan mode to navigate in a graph

Pan mode makes it easier to view different parts of a component graph that is larger than the display area in your browser. You can click and drag to quickly move the whole graph and see the portions that were outside the viewing area.

To enter Pan mode, click the Pan icon  in the toolbar.

About select mode

In Select mode, you can select a single component or click and drag to select multiple components.

To enter Select mode, click the Select icon  in the toolbar.

About groups

A group is a collection of components. To simplify a graph, group components, and then save the graph as a custom view. Groups have these characteristics.

- There are two types of groups: system-created and user-created.
- Groups can be nested; that is, a group can contain one or more other groups
- Groups can be collapsed or expanded
- Groups can be positioned anywhere on the graph.
- Groups can be ungrouped.

When you hover your cursor over a group, a tooltip window shows a summary of the group's contents.

About system-created groups in the system graph and custom views

When Detect generates a workspace graph, it groups some components automatically, as follows.

- Component group—If a data component is used by only one component, and is not used by any other component, it is placed in a component group. Component groups are white rectangles on the graph, like individual components, but they have a plus sign in the upper left corner that you can click to expand them.
- Trigger group—If there are multiple trigger systems in the workspace, each one is placed in a trigger system group. Trigger groups are white rectangles on the graph, like individual components, but they have a plus sign in the upper left corner that you can click to expand them.
- New components group—When you add components to a workspace, Detect creates groups in all of your custom views. Detect places the new components in a system-created group called New Components. This ensures that new components do not disrupt the layout in your custom view. You can then pull the new components out of the New Components group and place them in the desired location in each of your custom views. New components groups are ovals, and have a plus sign in the upper left corner that you can click to expand the group.

About user-created groups in a custom view

To create your own group, select the components you want to group, and then right-click to open a context menu where you can select the Group command.

When you create a group, you can name it and select a color. User-created groups appear as ovals on the graph.

About connectors

Detect displays different types of connectors between components and groups of components, based on the type of components that are connected. When you move components, their connectors remain attached.

Showing and hiding connectors

You may want to hide some or all connectors to reduce clutter caused by crossing lines. The connector buttons at the top of the Component Graph allow you to hide or show all connectors or to select groups of connectors to hide or show.

You can also right-click a component to see a context menu that allows you to show or hide the connector for a single component.

Connector types

Different relationships between components are reflected in the appearance of the connectors. Hover your cursor over the connector buttons at the top of the Component Graph to see tooltips that serve as a key to the connector types on the graph. The connector types are:

- Event generating connectors—Simple, Container Manipulators, FLI, BLI, Pattern, and Action components either generate events, receive an event, or do both. An event generating connector shows the connection between a component that generates an event and the component that consumes that event.
- Data flow connectors—A data flow connector shows the connection between components that do not generate events: Join, Select Function, Math, Qualifier, and Trend components.
- Use container connectors—Connects a component with a container whose data it reads.
- Modify container connectors—Connects a component with a container that it modifies.
- Select connectors—Connects a component to the Select Function component that it uses.
- Math connectors—Connects a component to the Math component that it uses.
- Qualifier connectors—Connects a component to the Qualifier component that it uses.
- Negative event generating connectors—Connects a Container Manipulator that uses the Negative (NOT) event of another Container Manipulator.
- Reset event connectors—Connects a component with a component whose event is used to reset it.

About the context menu

You can access the context menu for any component by right-clicking in the graph. The following table describes the commands available in the context menu. The options available vary, depending on where in the graph you right-click.

Commands	Description
Show connectors	Show or hide the connectors for the selected component.
Expand group	Available only for grouped components. Opens the group so you can see the components it contains.
Expand all in group	Available only for grouped components. Opens the group so you can see the components it contains. If the group contains other groups, opens these groups as well.
Collapse group	Available only for grouped components. Closes an expanded group.
Ungroup	Available only for grouped components. Separates all components in the group.
Ungroup all trigger groups	Available only in a system-created graph that has trigger groups, when you right-click on the graph's canvas. Expands all trigger groups.
Enter group	Available only for grouped components. Displays the expanded group and hides the rest of the graph.
Exit group	Available only when you have selected Enter group. Closes the currently expanded group. In nested groups, moves one level up in the group hierarchy.
Home	Available only when you have selected Enter group. Closes all levels of the currently expanded group and displays the top-level view of the graph.
Edit	When you right-click a component, this command opens the Component Editor for the selected component. When you right click a group, this command opens the Group properties window for the selected group.
Delete	Deletes the component and all of its dependent components.
Show component summary	Opens a window displaying the component summary.
Add new component	Opens a sub-menu that allows you to select one of the component types. When you select a component type, the component editor opens and you can create a new component.
To front	Where components overlap, displays the select component on top of other components.
To back	Where components overlap, displays the selected component under other components.
Select all	Selects all components. Available only when you right-click on the graph's canvas.
Fit to screen	Fits the whole graph in the visible area. Available only when you right-click on the graph's canvas.
Actual size	Shows the graph actual size. Available only when you right-click on the graph's canvas.
Print preview	Shows a preview of how the graph will print. Available only when you right-click on the graph's canvas.
Print	Prints the graph. Available only when you right-click on the graph's canvas.

Suggested method for creating a custom view

Typically, you create custom views to simplify and clarify parts of a complex trigger system within a workspace, as an aid to understanding it. This section suggests a general approach to accomplishing this task.

First, simplify the graph to understand the flow of logic in the trigger system:

1. Hide all connectors except event generating connectors.
2. Collect all data components into one or more groups and hide its connectors.

   The data components are:
   - Container
   - Join
   - Math
   - Select
   - Qualifier
   - Trend
3. Arrange the ungrouped components, which are the event generating components.

   Second, when you understand the logic flow, restore the important data components to the view:
4. Identify the key data components and ungroup them.
5. Turn on connectors to these data components.
6. Adjust the label size and length.
7. Name the view, give it a description, and save it.

About the Component List

The Component List tab in the Workspace Editor displays the workspace components in list form. The component list provides one of the two ways you can access the Component Editor to build trigger systems.

This section describes the features available on the Component List tab.

Action bar in the Component List tab

The action bar provides the functions available on the Components List tab within the Workspace Editor, as listed in the following table.

Feature	Usage
Filter	Select from a drop-down list of component types. The result is a filtered display, listing just the components of that type within the current workspace. Note: when you use filter (sort) the display is no longer in tree mode.
Add new component	Select from a drop-down list down list of component types to add a new component of the selected type to the workspace.
Copy from	Select the component to be copied, then click the Copy function. The Component Editor opens, pre-populated with the same properties as the component you selected to copy.
Delete	Select the component to be deleted, then click the Delete function.

Feature	Usage
Access administration	The Administer option is available only to users who have permission to Administer Component XML. This option can be used to view and edit the underlying XML source code for the component. It can also be used to unlock the component or change the owner of the component.
View usage	Select the component to be viewed, then click the View Usage function.

These features are described in more detail elsewhere in this guide.

Columns in the Component List tab

When you view the Component List in the Workspace Editor, the components within that workspace are shown in a hierarchical fashion to show the relationship between them.

The following table describes the columns displayed for each component.

Column	Description
⊞ and ⊟ parent/child indicators	If present, the plus and minus signs indicate parent/child relationships for components and trigger systems displayed in the list. For more information, see "Columns in the Component List tab."
Radio button	Used to select a specific component for a desired action, such as Copy or Edit.
 (information)	Clicking the information symbol for a component activates a Component Summary pop-up window that displays the component's name and a summary description for it.
Name	A short name for the component.
ID	A numeric identifier for the component.
Type	The component type.
Owner	The login name of the person who created the component, or the person assigned to it by the administrator.
Strategy	The strategy, if any, entered for the component.
Labels	The categories with which the component is associated The Detect system administrator creates the label headings and values, which may vary from installation to installation.

Sort by a column

Headings that are underlined indicate an available sort function.

All but **Hierarchy** are similar in the way they operate. For most of the columns, clicking on their header sorts the list by the values in that column (alphabetically or numerically, in ascending order). Clicking the header again reverses the order.

For the Hierarchy column, clicking the heading collapses the display of any expanded parent/child relationships.

Parent/Child Relationships Indicators

By default, the Component List displays a flat, non-hierarchical view of a workspace. To see components listed hierarchically with the action components at the top of the hierarchy, you can click the **Hierarchy** column. You can drill down into the hierarchy to see the sub-components on which a top-level component is dependent.

In the hierarchy view, trigger systems are displayed at their highest hierarchy level and are listed in ascending order by ID. Parent/child notation indicates the existence of underlying component relationships.

The plus and minus signs (⊞ and ⊟) indicate parent/child relationships for components displayed. The plus symbol associated with a component, or attached to a component type in the **View Usage** screen, indicates a parent with child components not currently displayed. Clicking the plus symbol of a parent relationship displays a nested view of the associated child (children).

The minus symbol indicates either a child with no child components or an expanded parent component.

Getting started with workspaces

Use the procedures in this section as an introduction to creating trigger systems in workspaces. For details about using components to build trigger systems, see the remainder of this guide.

Related tasks:

"To access the Workspaces area"

"To create a new empty workspace"

"To change the name or description of a workspace" on page 13

"To view trigger summary information for a workspace" on page 13

To access the Workspaces area

1. In the address bar of your Internet Explorer browser, enter

   `http://machine name/Detect`

   where machine name is the name of the machine on which the Detect web application is installed.
2. Disable popup blocking in the browser.

   Note: Pop-ups are used throughout the system. If you do not disable popup blocking, you will not be able to use all of the features.
3. Enter your user name and password, then click **Enter**.

   Detect opens in the Workspaces area.
4. To navigate to the Workspaces area from other areas of Detect, click **Workspaces** in the navigation bar.

To create a new empty workspace

1. Select **Workspaces** in the navigation bar.
2. Click the **Create new** icon  .
3. Enter a descriptive name for the workspace in the **Name** field.

4. Enter a unique, 3-character identity code for the workspace in the **Code** field.

   You will use this identity code when to refer to this workspace when you use other Detect features.

   Workspace codes are case sensitive, so T01 is not the same as t01.
5. Enter a description for the workspace.
6. Click **OK**.

   You can copy, clone, or move components into the new workspace, or create new components in it.

To change the name or description of a workspace

1. Select **Workspaces** in the navigation bar.
2. Click the name of the workspace for which you want to change the name or description.
3. Click the name of the workspace for which you want to change the name or description.
4. In the Properties tab of the Workspace Editor, change the name or the description.
5. Click **Save Changes**.

To view trigger summary information for a workspace

The trigger summary information for a workspace is listed on the workspace's Properties tab. The trigger summary lists the action components within the workspace. If there are descriptions on the components, it shows them as well.

1. Select **Workspaces** in the navigation bar.
2. Click the name of the workspace for which you want to see the summary.
3. In the Properties tab of the Workspace Editor, view the **Trigger Summary** area.

Deleting workspaces

When you delete a workspace, you also delete all of the components within it. If a component is a copy of one that exists in another workspace, then only the copy in this workspace is deleted.

If a workspace is used by a runset, references to that workspace are removed from the runset.

Note: Workspace deletion cannot be undone.

To delete workspaces from the Workspaces manager

You can delete multiple workspaces at once from the Workspaces manager.

1. Select **Workspaces** in the navigation bar.
2. Select the checkbox for the workspace or workspaces you want to delete.
3. Click the delete icon

   in the upper right corner of the Workspaces area.
4. Click **OK** in the confirmation window.

To delete a workspace from the Workspace Editor

1. Select **Workspaces** in the navigation bar.
2. Click the name of the workspace you want to delete.
3. Click **Delete**.
4. At the confirmation window, click **OK**.

Creating and managing runsets

Use the procedures in this section to create and manage runsets.

To create a runset

1. Click **Runsets** in the navigation bar.
2. Click **New**.

   The Edit Runset window opens.
3. Complete the fields as follows.
   - Enter a name for the runset in the **Name** field.
   - Enter 3-character code for the runset in the **Runspace Code** field.

     Runspace Codes are case sensitive, so T01 is not the same as t01.

     Choose a runspace code that is different from all other runspace codes in your system. This code is used to identify the outcome table associated with this runset.
   - Enter a description for the runset in the **Description** field.
   - Select components to which the runset refers according to the directions in "To select the trigger systems for a runset."
4. Click **Save** to save the new runset and close the Edit Runset window, or click Close to cancel without saving.

To select the trigger systems for a runset

You can include the following types of items in a runset.

- Action components in workspaces
- Container manipulators in workspaces
- Other runsets

1. To include actions, select **Actions** from the **Type** list. Repeat the following procedure for each workspace from which you require actions for the runset.
   a. Select a workspace containing required actions from the runspace codes listed in the **Workspace** list. The actions in the selected workspace are listed in the Available Components pane.
   b. Select one or more actions from the list, and click the right-arrow button. The actions you select to be referenced by this runset appear in the Selected Components pane.
2. To include runsets, select **Runsets** from the **Type** list. Select one or more runsets from the list, and click the right-arrow button. The runsets you select to be referenced by this runset appear in the Selected Components pane.
3. To include container manipulators, select **Container Manipulators** from the **Type** list. Repeat the following procedure for each workspace from which you require container manipulators for the runset:

a. Select a workspace containing required container manipulators from the runspace codes listed in the **Workspace** list. The container manipulators in the selected workspace are listed in the Available Components pane.
b. Select one or more container manipulators from the list, and click the right-arrow button. The container manipulators you select to be referenced by this runset appear in the Selected Components pane.

4. You can review all of the components you select to be referenced by the runset in the **Selected Components** pane of the Edit Runset window. If you need to remove any items from this list, select the items, and click the left arrow.
5. Click **Save** to save the runset.

To edit a runset

1. Select the runset name from the Runset Manager.

   The Edit Runset window opens.
2. Edit the runset content according to the directions in "To select the trigger systems for a runset" on page 14.

To delete a runset

Click **Delete** from the Runset Manager to delete a selected runset.

Note: If you delete a runset that is referenced by other runsets, those references are removed.

Chapter 4. Component Basics

IBM Unica Detect functions by recognizing transactions that trigger events. Trigger systems contain the logic for recognizing transaction patterns and events, and the building blocks used to create the trigger systems are called components. Components are grouped within a workspace to create one or more a trigger systems.

Types of components

This section provides a high-level overview of the component types that you can use to build trigger systems, and provides a brief summary of their use.

Component Type	General Function	Description
Backward Inactivity	Can trigger actions	A component that is triggered by the absence of a specific activity preceding a specific activity. A behavior that is defined by recognizing the occurrence of a specific event and then looking back in time (a defined time period) to see whether another specific event has occurred. The Backward Inactivity component fires if the event being "looked for" did not occur. **Example:** Trigger an event if event A happens and event B did not happen in the specified prior time frame. **Example:** Trigger an event when a customer makes use of the ATM after more than 1 month of not using any teller services. Also known as Backward Looking Inactivity and BLI.
Container Manipulator	Can trigger actions	A definition of a process used to manipulate the data in containers. Add, delete, and manipulate contents of Containers. Can trigger events based on the content of Containers. Do clauses include: insert, delete, and soft insert. **Example:** When the dollar amount of product a customer purchased in one month exceeds $200 (value in the container), offer a discount.

Component Type	General Function	Description
Forward Inactivity	Can trigger actions	A component that is triggered by the absence of usual activity. A behavior that is defined by recognizing the occurrence of a specified event and waiting 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:** Trigger when a web-trade customer, who usually trades once a month, does not trade for two consecutive months. **Example:** Trigger an event if event A happens and event B does not happen in the time frame specified in the future. Also known as Forward Looking Inactivity and FLI.
Pattern	Can trigger actions	A group of events occurring within a defined time frame. **Example:** Trigger when customer uses his credit card four times a month during a three month period and is a platinum card holder.
Simple	Can trigger actions	A defined set of business-relevant conditions that recognize certain attributes in a particular transaction. Trigger an event if the specified conditions are satisfied. **Example:** Credit card purchase over $5000 or an international phone call to Italy
Action	Writes outcomes to the Outcome database	A component that triggers an outcome message (to the external system) based on criteria being met on a trigger system. Defined pieces of data (customer#, time, reason to contact) that are written to the Outcome database when a specified event occurs. Could represent a recommendation to contact a customer due to a pattern of behavior. **Example:** Send coupon for free 10 minute international call when customers who have GOLD status complete 15 international phone calls in one month.

Component Type	General Function	Description
Join Function	Returns results from an operation on a database table	A definition of a process that joins two record-set objects. It returns another record set whose type structure has been pre-defined using the Type Descriptor. **Example:** Join the container holding customer purchases including store number with the look-up table containing information about all stores which contains store numbers and regions in order to determine the region of each customer purchase in the container **Example:** Join the container holding stock trades including the stock CUSIP number with the look-up table containing information about all stocks which contains CUSIP numbers and industry sectors in order to determine the industry sector of each stock purchase in the container **Example:** Join the banking product code on an incoming transaction to a container containing all products the customer has. If the result contains a line in the join, then the customer has this product. If it contains no lines then the customer does not have the product, and hence this transaction represents a new product for this customer.
Select Function	Returns results from an operation on a database table	A definition of a process that reduces the information in a table according to some specified criteria.
Math	Used internally by other components	A defined mathematical formula that calculates a numeric result based on numeric data from one or more data sources and can be used in Simple Event components to refine the event which must occur. **Example:** Credit card purchase over $5000 (Simple Event component) where the purchase amount is 95% of the available account balance (refining mathematical formula.)
Container	Used internally by other components	An area used to store data from transactions or a profile. Holds data that can be collected, used in calculations, and tested by a Container Manipulator. **Example:** Save the dollar amount of products a customer purchased in one month.

Component Type	General Function	Description
Qualifier	Used internally by other components	Customer attributes that help define the selection criteria of whether the component should trigger or not. A defined set of customer characteristics that reveal a quality about the customer as opposed to recognizing the behavior of that customer. Qualifier components act as filters, refining the firing of Simple Event components and Action components to those customers who posses the defined quality. **Example:** GOLD customer is part of the action component described at the beginning of this table, and is a Qualifier.
Trend	Used internally by other components	component that simplifies the system's ability to detect trends and unusual transactions. It can detects trends, significant spike transactions, or transactions that exceed standard deviations. **Examples:** The average monthly balance has been increasing by 10% over the last six months, a deposit is 25% larger than any other deposit made in the last 6 months, or a balance that is more than two standard deviations above the monthly average for the past year.

The following lists show the components grouped by their function.

Components that can trigger events

- Backward Inactivity
- Container Manipulator
- Forward Looking Inactivity
- Pattern
- Simple

Components that write outcomes to the Outcome database

- Action

Components that return results from an operation on a database table

- Join
- Select

Components that are used internally by other components

- Math
- Qualifier
- Trend
- Container

Administering components

In some cases you may want to perform advanced administrative functions on a component. For example, you may need to unlock a component that is in use by a user, change the owner of a component, or work directly with the component's XML. The **Administer** action is available to users who have the Administer XML permission. For more information, see the chapter, Administering Components.

Copying components and workspaces

If you want to reproduce the functionality of an existing workspace, you can copy an existing workspace or individual component and use it as the basis for a new workspace or component.

You can copy a component from a different workspace or copy a component within a workspace.

You can also copy an entire workspace, including all of its components.

If you modify a copy, the original is not changed.

When you copy components, you create exact replicas of them. Copied components all have the same ID as the original.

Best practice: do not change copied components used in the same runset

Using copied components in the same runset has a performance benefit. Runsets process copied components more efficiently than identical components that are created separately or cloned.

When multiple components in a runset have the same ID, (as copied components do) the runset discards all but one copy. This results in faster processing. However, to avoid unexpected results, you must take care not to change the definition of any of the copied components.

It is also a best practice to keep track of those components that you copy, so you do not inadvertently change a copy. You should create a workspace named COM (for Common) that contains all the components you copy, and that you use COM as a prefix in all the copies of components, to make it easy to recognize that these are copied components that should not be changed.

To copy components from one workspace to another

1. Select **Workspaces** in the navigation bar.
2. Click the name of the workspace into which you want to place the copied components.
3. Select the Copy Components from tab.
4. From the drop-down list, select the workspace that contains the components you want to copy.
5. Click **Copy components**.
6. Expand the hierarchy of components.
7. Click the check the box next to each component that you want to copy.

   To select or deselect all of them, click the checkbox in the header bar at the top of the component list.

8. Click **Submit** at the bottom of the screen.

   If there are components dependent on the ones you select, Detect copies them too and tags the selection mode as **Dependent Selection**.
9. Click **Commit Copy**.

To copy components within a workspace

1. Select **Workspaces** in the navigation bar.
2. From the list of source workspaces, click the name of a workspace.
3. Select the Component List tab.
4. Select the component to be copied.
5. Click **Copy** on the Actions bar.
6. At a minimum, change the name of the component under the **Properties** tab and click **Save**.

To copy an existing workspace with its components

1. Select **Workspaces** in the navigation bar.
2. Click the checkbox next to the workspace you want to copy.
3. Click the **Create new from**  icon.
4. Enter a descriptive name for the workspace in the **Name** field.
5. Enter a unique, 3-character identity code for the workspace in the **Code** field.

   You will use this identity code to refer to this workspace when you use other Detect features.

   Workspace codes are case sensitive, so T01 is not the same as t01.
6. Enter a description for the workspace.
7. Click **OK**.

   A new workspace is created, populated with copies of all of the components contained in the original.

Cloning components

When you clone components, copies of components that exist in one workspace are added to another workspace. Unlike copies, the new cloned components have new unique IDs and new component IDs. Reasons to clone components include the following.

- You want to modify the new components.
- You want to make a test workspace with some changes.
- You want to make a new workspace based on the logic of another workspace.

If you modify a clone, the original is not changed.

To clone components in one workspace to use in another

1. Select **Workspaces** in the navigation bar.
2. From the list of source workspaces, click the name of the workspace into which you want to place the components you clone from another workspace.
3. Select the Copy Components from tab.
4. From the drop-down list, select the workspace that contains the components you want to clone.

5. Click **Clone the components**.
6. Expand the hierarchy of components.
7. Click the check the box next to each component that you want to clone. To select or deselect all of them, click the checkbox in the header bar at the top of the component list.
8. Click **Submit** at the bottom of the screen.

   If there are components dependent on the ones you select, Detect clones them too and tags the selection mode as **Dependent Selection**.
9. Optionally, use string substitutions to tailor the name, description, and label 2. All the components in the cloned workspace get new IDs.
10. Click **Commit Clone**.

Moving components

When you move components, they are removed from the source workspace and added to your target workspace.

To move components from one workspace to another

1. Select **Workspaces** in the navigation bar.
2. Click the name of the workspace into which you want to move the components.
3. Select the Copy Components from tab.
4. From the drop-down list, select the source workspace that contains the components you want to move.
5. Select **Move components**.
6. Expand the hierarchy of components.
7. Click the check the box next to each component to be moved. To select or deselect all of them, click the checkbox in the header bar at the top of the component list.
8. Click **Submit**.

   If there are components dependent on the ones you select, Detect moves them too and tags the selection mode as **Dependent Selection**.
9. Click **Commit Move**.

Adding and editing components

You can add a component from the Component Graph tab or the Component List tab of a workspace. Detailed information about creating each component type is provided elsewhere in this guide.

To add a component to a workspace

1. Access the Workspace Editor for the workspace to which you want to add a component.
2. Select the Component List tab or the Component Graph tab.
3. Select the component type from the **Add new component** drop-down list.

   The Component Editor for that component type opens.

   See “Types of components” on page 17 for a description of the component types.
4. Specify the properties of the component and click **OK**.

To edit a component

1. Do one of the following.
   - On the Component List tab, click the name of the component to be edited.
   - On the Component Graph tab, right-click the component and select Edit.

   The Component Editor for the component opens.
2. Modify the component and save.

Deleting workspaces

When you delete a workspace, you also delete all of the components within it. If a component is a copy of one that exists in another workspace, then only the copy in this workspace is deleted.

If a workspace is used by a runset, references to that workspace are removed from the runset.

Note: Workspace deletion cannot be undone.

To delete a component from a workspace

1. Access the Workspace Editor for the workspace that contains the component that you want to delete.
2. Select the Component List tab or the Component Graph tab.
3. Do one of the following.
   - In the Component List tab, select the component and click the **Delete** icon  .
   - On the Component Graph tab, right-click the component and select Delete from the context menu.

   You see a warning if any components are dependent on the selected component. If there are, they will also be deleted.
4. Click either **Continue** or **Cancel**.

About the Component Editor

You use the Component Editor to specify the properties of components. The Component Editor is a window with two areas, each with distinct functions as described below:

- A set of tabs located in the left half of the screen, where you can specify new properties for the component. These vary by component type.
- A component summary area located in the right half of the screen, where you can view a summary of the properties selected for the current component. If you click a link in the summary area, you can view and edit the details of existing properties.

About component properties

All components have properties, which you set on the Properties tab. From the component's Properties tab, you can change:

- the name of the component
- the description of the component

- the strategy, start and end dates, and period mode
- the trigger controls
- the label selections

The following table describes the fields that may comprise a component's properties. Some components have a subset of these fields.

Field	Description
Name	Required field with a maximum length of 36 characters.
Description	Optional field, but it is recommended that you give the component a good description.
Strategy	Optional field to aid in categorizing the component. The default is **NONE**.
Start Date	Date the component becomes active. The component is activated at 12:00 am of the start date. Default value is 1/1/1980 which means the component will be active immediately.
End Date	Date the component is considered expired and will no longer be used in the application. The component will expire at 11:59 pm of the end date. The default is 12/31/2079. NOTE: Start and End Dates can be modified at any time.
Period Mode	Refines the triggering time of the component such as weekly, monthly, quarterly, or yearly. For examples, see Period mode examples.
Trigger Control	Trigger Control settings specify the maximum number of times and the frequency at which the component can trigger for a given customer. For details, see About trigger control settings.
Labels	Optional fields to aid in categorizing the component. The label headings and dropdown choices you see are defined by the System Administrator.

Period Mode examples

Period Mode	Period Start	Period Length	Example Description
none			The component will run between the dates set.
weekly	Pick a day of the week. For example, **Tuesday**.	Pick the period length. For example, **End of period**. End of period for a week is Saturday.	The component will run in weekly increments, from the start date through the end date. The component in this example will only trigger on events that occur between Tuesday and Saturday. It will not fire based on an event that happens on a Sunday or a Monday.
monthly	Pick the day of the month for the component to start. For example, **5**.	Pick the period length. For example, **Full period**.	The component will run in monthly increments, from the start date through the end date. The component in this example will trigger on events that occur between the 5th of the month through the 4th of the next month.

Period Mode	Period Start	Period Length	Example Description
quarterly	Pick a day within the quarter for the component to start. You can pick the numeric value for any day within the quarter. For example, **1** to start on the first day.	Pick the period length. You can pick **Full period** or **End of period** or any day of the quarter to end. For example, **80**.	The component will run in quarterly increments from the start date through the end date. The component in this example will only trigger on events that occur between the first and through the 80th day of the quarter.
yearly	Pick the day of the year for the component to start. For example, **15**.	For the period length, pick the day of the year to end on. For example, **200**.	The component will run in yearly increments, between the start date and end date. The component in this example will only trigger on events that occur between the 15th day of the year through the 200th day of the year.

About trigger control settings

The trigger control (trigger limit) includes the ability to reset trigger limits on components. This functionality provides a way to control the number of times a component fires for a customer (ID) over a given period of time. The value for the time quantity can be derived from a data source value, constant, or math expression.

With the trigger controls, you can specify behavior on an individual component basis. For example you can select a point in time for the trigger control to re-set it self. You can create a rule with a trigger limit of one for a period of one day. If the rule fires more than once within a day, the engine will suppress the firings of the triggers. The component will re-set it self after the allotted time frame has expired.

Components that use trigger controls

These trigger controls are available in the following editors:

- Action
- Container Manipulator
- Pattern
- Simple Event
- Backward Looking Inactivity (BLI)
- Forward Looking Inactivity (FLI)

Trigger control options

Four trigger control setting options are available:

- **Always**: The component always fires.
- **No more than n times**: The maximum number of times the component can fire over its lifetime.
- **No more than n times in a cycle**: The maximum number of times the component can fire during one cycle. (A cycle is one day's worth of feeds.)
- **No more than n times in...**: The maximum number of times it can fire during a certain interval of time. Any component reaching its firing limit is suppressed

for the rest of that time interval. For example, you can set a maximum number of three firings over the course of two days.

The options change depending on whether you choose Data Source Field, Literal, or Math Expression from the first drop-down list. The length of time options are:

– Day
– Calendar Week
– Calendar Year
– Calendar Month
– Rolling Week
– Rolling Month
– Rolling Year

When Calendar Year is the interval, a component that reaches the limit will be eligible to fire again starting at midnight of January 1st of the next year. If the component reaches its limit on midnight of January 1st, it will be suppressed for a whole year. If the component reaches its limit on 11:59:59 on December 31st, it will be suppressed for only 1 second.

About functions

Functions perform some additional processing on the data returned by the component.

Components that use functions

Functions are available in the following component types.

- Math
- Qualifier
- Container Manipulator

Function definitions

The functions are defined as follows.

- Sum–total the values of the data.
- Minimum–use the lowest of the values.
- Maximum–use the highest of the values.
- Average–use the average of all the values.
- Count–count the number of pieces of data.
- Count Distinct–count the number of pieces of unique data.
- Mode–use the value of the most frequently occurring piece of data returned by the component, when compared to all the pieces of data returned by the component. When a tie situation is encountered in determining the most frequently occurring piece of data, an age factor is used. The data that is the oldest is the one returned by the Mode function.
- Standard Deviation–the measure of the scatter of values around the average of the data values returned by the component.
- First–use the value with the earliest timestamp.
- Last–use the value with the most recent timestamp.
- Data —all the saved data

The Time Constant field type

The Math, Qualifier, and Forward Looking Inactivity components require you to select a field type. If you select Time Constant as the field type while you are building a component, you have the options shown in the following tables.

The following time constants return a time stamp, in units of days.

Time Constant	Definition
Beginning of Next Month	Midnight of the first day of the month following the current month.
Beginning of *N* Months Ago	There are 12 of these options, from 1 month prior to the current month to 12 months prior to the current month. Midnight of the first day of the designated month.
Beginning of the Current Day	Midnight today.
Beginning of the Current Month	Midnight of the first of this month.
Beginning of the Current Quarter	Midnight of the first day of the quarter.
Beginning of the Current Week	Midnight of Sunday.
Beginning of the Current Year	Midnight of January 1 of the current year.
Current TimeStamp	Now. For example, in a math expression, Current Timestamp - 7 is exactly one week ago.

The following time constants return a numeric value.

Time Constant	Definition
Current Day (Day of the Month)	1 – 31
Current Month	1 – 12
Current Quarter	1 – 4 (Q1 begins Jan 1)
Current Year	Current year, four digits

Chapter 5. Comparison Clauses

Both the Simple Event and Qualifier components use comparison clauses.

In its simplest form, a comparison clause compares a specified field from the data source with a specified constant value or a field from a data source. You can also nest comparison clauses to build complex comparisons.

You can compare data from a transaction or database, math operations, constant values, or Qualifier components.

Note: Simple Event components have some restrictions on the data it can compare. See Chapter 7, "Simple Event Components," on page 39.

Examples of operands in comparison clauses include:

- a data source field value from a transaction data source with a field value from a profile data source
- two data source field values from a profile data source
- a data source field value with a constant
- a data source field value (from a transaction data source or profile data source) with a value calculated in a Math component
- the contents of a Container in a Math component

About logical operators

Comparison clauses use logical operators to develop the logic. The following table describes each operator.

Operator	Definition	Example
and	All operands must be true for the component to trigger	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 The component will trigger when all three conditions are true.
or	Any of the operands must be true for the component to trigger	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. The component will trigger when any one of the conditions, any combination of the conditions, or all of the conditions are true.
nand	"Not and" - any one of the operands must be false for the component to trigger	Look for transactions that do not have a price per share greater than $99.99 OR were not processed on June 15, 2001 OR were not for accounts in New York. The component will trigger when any one of the conditions, any combination of the conditions, or all of the conditions are false.
nor	"Not or" - all of the operands must be false for the component to trigger	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. The component will trigger when all of the conditions are false.

About comparison operators

The comparison operators available in comparison clauses are:

- Less than
- Less Than or Equal To
- Equal
- Not Equal (Simple Event components only)
- Greater Than
- Greater Than or Equal To
- Like
- Is Member Of (Qualifier components only)

About the Like comparison operator

The Like comparison operator looks for records that contain information that is similar to the regular expression that you define as the value.

For example, you could use the qualifier to help find store names that contain the word Discount or customer IDs that begin with the number 9.

To find the word "Discount" you would type the string Discount into the dialog box so that it searches for the word Discount anywhere in that field. To find a string in a particular position in the field, you must use more sophisticated regular expressions.

When you use the Like comparison operator, Detect does not validate the value that you enter. Refer to published information about regular expressions to help you build or test the syntax for the regular expression you set as the value. For example: `http://www.regular-expressions.info/reference.html` for building the expression or `http://regexlib.com/RETester.aspx` for testing it.

About the Is Member Of comparison operator

The Is Member Of comparison operator is available only in the Qualifier component.

The Is Member Of feature provides an efficient method for identifying and acting on large quantities of lookup table data to determine which records match a specified value. Qualifier components that use Is Member Of can evaluate only the results of a Select function. You can use the Is Member Of feature to determine if a value is member of the result of a Select qualifier.

Example of why you may want to use the Is Member Of function

Suppose that you offer 1000 products for sale. Of the 1000 products, there are 300 for which you send a rebate coupon when a customer purchases a certain quantity.

On first thought, you might create a Simple Event that recognizes each one of the 300 product numbers. But, this method creates a few problems:

- A Simple Event that identifies 300 product numbers is cumbersome to create.
- A Simple Event that identifies 300 products is difficult to maintain should you need to change the products included in the promotion.

- There is a limit to the number of characters that can be included in every component. The 300 product codes that need to be identified may exceed the length boundary of the Simple Event you try to create.

Detect provides the Is Member Of feature to effectively address these issues.

Example of how to use the Is Member Of feature

The following steps describe an efficient method of identifying the 300 products in the example described in "About the Is Member Of comparison operator" on page 30.

1. Create a lookup table listing all 1000 product codes and a Y/N indicator for whether the product offers a rebate.

   Product codes can easily be added to this table (or deleted), and the associated Y/N indicator can be changed as products are included or removed from the promotion.
2. Create a Select Function, based on the lookup table, where the value in the Y/N indicator equals Y. Include the product code in the Outcome Type Descriptor.

   This step efficiently identifies only the products that are included in the promotion.
3. Create a Qualifier component that determines if the product code in the transaction feed record is a member of the list of product codes identified by the Select Function.

   You can do that by adding a comparison phrase to determine whether the product code on the incoming data feed is a member of the product code field of the Select you defined in the previous step.

About data source types in comparison clauses

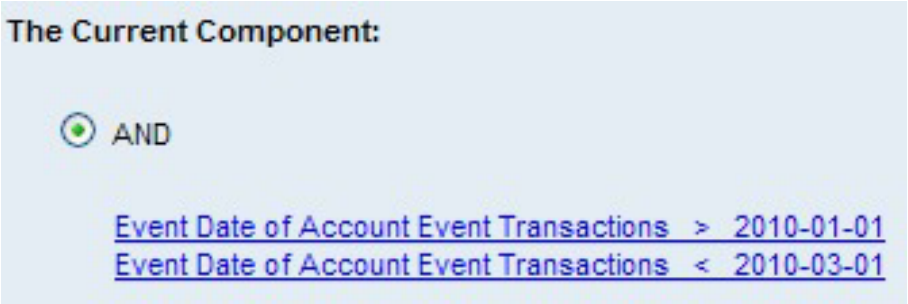
As you build comparison clauses, you select or enter the data to compare. Your choices differ depending on whether you are specifying a Simple Event or Qualifier component, and on the data type you select.

- When you select a Container or Select Function component as the data type, you can apply a function that performs some additional processing on the data. See About Functions for details.
- When you select a Time Constant as the data type, you can choose from a number of time-related options. See About the time constant data source type for details.

Building comparisons

You build comparison clauses in the Details tab of the Component Editor.

- Use the left pane to select the comparison clause operators and the data to be compared (operands).
- The right pane displays the operators and operands you selected in a comparison hierarchy. The operator is listed before any of the operands. For example, a clause whose logic is "Look for transactions whose date is later than January 1, 2010 AND earlier than March 1, 2010" would look like this in the summary area:



- To edit a clause, select it in the Summary area make your edits in the left pan
- The **Add Operator up a level** checkbox is available when the highest operator of a component is selected in the summary area. Selecting this checkbox applies a new operator and its clauses to a level higher than the first operator of the component. The new operator becomes the highest-level operator, placing the newly included operator at the highest level.
- The **Add Operator with the following clause** checkbox appears only if you have added an operator with no clause.

Example of building a simple comparison

The following procedure builds a simple comparison clause with a logical structure like the following:

Look for transactions with a value greater than $100

AND

with a date later than January 1, 2010

it looks like this in the Summary area:

To build a simple comparison clause

1. Select an operator AND in the **Operator** drop-down list.
2. Select the **Add Operator with the following clause checkbox**.
3. Build the first half of the AND clause and click **Add**.
4. Build the second half of the clause and **Add**.

   Note that the **Add Operator with the following clause** checkbox remains selected.
5. Click **OK** to save.

Example of building a nested comparison clause

The following procedure builds a nested comparison clause with a logical structure like the following:

Look for:

an account with automatic bill payment enabled AND that is a high value account

OR

an account with a HELOC line limit greater than or equal to $100,000 AND that is a high value account

It looks like this in the Summary area:

To build a nested comparison

1. Select OR in the **Operator** drop-down list.
2. Clear the **Add Operator with the following clause** checkbox.
3. Click **Add**.

   The OR operator is added in the Summary area with no clauses.
4. Select AND in the **Operator** drop-down list.
5. Select the **Add Operator with the following clause** checkbox.
6. Build the first half of the AND clause and click **Add**.
7. Build the second half of the AND clause and click **Add**.

   Note that the **Add Operator with the following clause** checkbox remains selected.
8. Select OR in the Summary area.
9. Verify that AND is selected in the **Operator** drop-down list.
10. Verify that the **Add Operator with the following clause** checkbox is selected.
11. Build the first half of the AND clause and click **Add**.
12. Build the second half of the AND clause and click **Add**.
13. Click **OK** to save.

Comparison clause examples

This section provides additional examples of using different operators in various combinations.

Example 1. One operator—one comparison hierarchy level

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, 2009

Example 2. Two operators—two comparison hierarchy levels

Look for:

accounts that are from New York AND had transactions processed on June 15, 2009

OR

Look for accounts that are from New Jersey AND that had transactions processed on June 16, 2009

OR

Look for accounts that are from Connecticut AND that had transactions processed on June 17, 2009

Example 3. Two operators—two comparison hierarchy levels

This example shows an AND comparison comprised of embedded OR comparisons.

Look for:

accounts from New York OR from New Jersey OR from Connecticut that had transactions processed on June 15, 2009

AND

the price per share was greater than $99.99

Example 4—One operator—NAND

This example illustrates the use of the NAND logical operator.

Look for transactions:

where the price per share was less than $99.99

OR

that are associated with accounts that are not from New York

OR

that did not process on June 15, 2009

Example 5—Two operators—T

Example 5—One operator—NAND

This example illustrates the use of the NAND logical operator to exclude transactions with specified criteria.

Exclude transactions:

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, 2009

Example 6—Two operators—two comparison hierarchy levels

This example illustrates the use of NOR with AND. NOR was added without any clauses, and then AND was added with clauses.

Look for:

accounts that are not from New York OR transactions that did not process on June 15, 2009

AND

accounts that are not from NEW Jersey OR transactions that did not process on June 16, 2009

AND

accounts that are not from COnnecticut OR transactions that did not process on June 17, 2009

Chapter 6. Qualifier Components

Qualifier components define specific characteristics of a customer, account, or household. For example, characteristics used in Qualifier components often include gender, state of residence, account types, customer status, social security number, or assets.

A Qualifier component evaluates to true when the referenced data values satisfy all the logical conditions defined in the component. Qualifier components never fire on their own. They are used internally by other components.

Uses for the Qualifier component

Here are some examples of ways to use Qualifier components.

- Apply a Qualifier component to a Simple Event or Action component to narrow triggering to a select population.
- Use a Qualifier component to add more complex conditions to a Container Manipulator, such as conditions that represent nested combinations of AND and OR operators.
- Include other previously defined Qualifiers as one of the required conditions in a Qualifier.

Sources of Qualifier component data

The characteristics defined in a Qualifier component can either be static data, which are independent of transactions, or they can be calculated from transaction activity.

A Qualifier can reference data from:

- A profile file (a delimited feed file)
- Database tables
- Container components
- Math components
- Select function component

Adding a Qualifier component

Add a new Qualifier component as described in "Adding and editing components" on page 23.

You use the following three tabs to build a Qualifier component.

- Details—This is where you build comparison clauses. See Chapter 5, "Comparison Clauses," on page 29.
- Qualifier—The Qualifier tab allows you to optionally select from the list of previously defined Qualifier components and include it as part of the Qualifier component's comparison clauses. Including a Qualifier lets you filter events at the initial transaction level. See the procedure in this section.
- Properties—The Properties tab is where you specify the name and other information about the component. See "About component properties" on page 24.

To include a Qualifier component within another component

The Qualifier tab allows you to select from the list of previously defined Qualifier components and include it as part of another component.

1. On the Qualifier tab, select a logical operator.
2. Select the Clause type from the **Type** drop-down list.
3. Select the component.
4. Click **Add** to add the Qualifier to the component.

Chapter 7. Simple Event Components

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

Simple Event components watch for the occurrence of a single transaction with attributes that meet specified criteria. A Simple Event component triggers an event when the criteria are met. When the Simple Event is triggered, Detect notifies all of the previously defined, higher level patterns and actions that are looking for that particular event.

Comparison clauses in Simple Events

In its most basic form, the Simple Event component uses a comparison clause to compare one field from a transaction data source to a specific 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 value *ACH Credit*.

For more about comparison clauses, see Chapter 5, "Comparison Clauses," on page 29.

Single transaction data source requirement

A Simple Event component must contain at least one comparison to a transaction data source field. Without a transaction the Simple Event would never fire. You can use transaction data from only one data source in a Simple Event component.

Using Qualifiers to refine a Simple Event

To include criteria from more than a single transactional data source in a Simple Event component, you can include a previously defined Qualifier component in the Simple Event component. Qualifier components are not restricted to a single data source.

About Math components in Simple Events

Take extra care when you use Math components in Simple Event components. Note the following considerations.

- A transaction data feed must be specified in either the Simple Event component or the Math component. Note that the Component Editor does not verify the contents of the Math component. If a transaction data feed is neither specified in a comparison clause nor in the associated Math component, the Simple Event component will trigger constantly.
- The one necessary parameter from a transaction data source can be deeply embedded, residing only in the Math component itself.
- When a Simple Event component includes a Math component that uses transaction data and also uses transaction data elsewhere in a comparison clause, the data must come from the same data source. If the two are different, the Simple Event component will never trigger, because Simple Event components look at only one transaction at a time.

Adding a Simple Event component

Add a new Simple Event component as described in "Adding and editing components" on page 23. Specify the component properties as described in the following procedure.

You use the following three tabs to build a Simple Event component.

- Details—This is where you build comparison clauses. See Chapter 5, "Comparison Clauses," on page 29.
- Qualifier—The Qualifier tab allows you to optionally select from the list of previously defined Qualifier components and include one or more Qualifiers as part of the Simple Event component's comparison clauses. Including Qualifiers lets you filter events at the initial transaction level. See the procedure in this section. See "To include a Qualifier component within another component" on page 38.
- Properties—The Properties tab is where you specify the name and other information about the component. See "About component properties" on page 24.

Chapter 8. Math Components

The purpose of a Math component is to provide a calculated numeric result that can be used as a parameter in other components.

Math components do not store the results of the calculated data; they return values used by other components. You can use Math components in the following component types.

- Simple Event
- Qualifier
- Container Manipulator

Math components do not trigger events.

Sources of Math component data

Math component can reference numeric data from the following sources.

- Data Source components
- Select Function components
- Container components
- Other Math components

Adding a Math component

Add a new Math component as described in "Adding and editing components" on page 23. Specify the component properties as described in this section.

You use the following two tabs to build a Math component.

- Details—This is where you build the math expression. See the procedure in this section.
- Properties—The Properties tab is where you specify the name and other information about the component. See "About component properties" on page 24.

To build a math expression in a Math component

1. In the Fields area of the Math Component Editor, make selections to build the first operand in the Math expression.

   To get started, select the a data type from the **Type** drop-down list.

   The remaining fields change based on the field type you select, as described in "Options available for Math components" on page 42.
2. Click **Add** so the resulting value becomes an operand in the Math expression.

   As you build the math expression in the Component Editor, the expression appears in the Expression box.
3. Continue building the expression.
   - Use the calculator keypad in the Constants/Operators area of the editor.
   - You can add constants (numerals) and operators (+, -, parentheses, etc.) to the display by clicking on their buttons.
   - Parentheses "()" control the order in which operators evaluate the operands.

- Use the MOD (modulus) operator button (shown as MOD in the expression) when you need to calculate the value of the remainder of a division operation. For example, if you want to validate that a transaction amount is in ten dollar increments, you could create a math expression like this: TranAmMOD 10.00

  Then you could use the Math expression in a Simple Event that compares the value created by the Math expression to zero. All transaction amounts in even ten dollar increments will result in a 0 remainder. All other transaction amounts will result in a remainder greater than 0.

4. Click **Save** to save the Math Expression component.

Options available for Math components

This section describes the fields available in the Fields area of the Math component editor.

Options in the Fields area

If you select **Container**, available fields are:

- **Name** - Select an existing container.
- **Field** - Select a field in that container.
- **Function** - Select a function. See "About functions" on page 27

If you select **Data Source**, available fields are:

- **Source** - Select an existing data source.
- **Field** - Select a field in the data source.

If you select **Math**, the available field is:

- **Expression** - Select an existing math expression.

If you select **Select Function**, available fields are:

- **Name** - Select an existing select function.
- **Field** - Automatically displays the data type.
- **Function** - Select a function. See "About functions" on page 27.

If you select **Time Constant**, the available field is:

- **Constant** - Select an existing time constant.
  See "The Time Constant field type" on page 28.

Chapter 9. Pattern Components

A Pattern component tracks combinations of events over a defined period of time, for a given customer, guided by a defined pattern. The defined pattern controls whether it is necessary for all the listed components to trigger in a specific order, or for all the listed components to trigger in any order, or for just one of the components to trigger.

A Pattern component requires a defined period of time during which the specified events must be recognized. For example, a printer purchase AND an ink cartridge purchase in 1 week.

Events that Pattern components can recognize

Pattern components can recognize events triggered by any of the Detect components that can trigger an event.

- Simple Event
- Backward Looking Inactivity
- Forward Looking Inactivity
- Container Manipulator
- Other Pattern components

Pattern components can also recognize artificial transactions. An artificial transaction is a system-generated event, created during run-time, that is based on an optional parameter of the engine process. Examples of artificial transactions are: End of Run End of Day Transaction.

Examples of patterns

Pattern parameters specify the combination in which the events must be recognized. Here are some examples.

- Any one of specified events.

  For example, purchased a printer OR purchased printer paper OR purchased an ink cartridge.
- All of the specified events.

  For example: purchased a printer AND purchased printer paper AND purchased an ink cartridge.
- All of the specified events in the defined order.

  For example, purchased a printer FOLLOWED BY purchasing printer paper FOLLOWED BY an ink cartridge purchase
- All of the specified events in the defined order, multiple times.

  For example:
  - a printer purchase (3 times)
  - OR a printer purchase FOLLOWED BY a printer paper purchase FOLLOWED BY an ink cartridge purchase. (This sequence repeated 3 times)

About reset events

You can optionally designate any event as a reset event. You should use the reset option only with events that must occur in sequence. The occurrence of the specified reset event removes from the Pattern component any recorded occurrences of required events. The Pattern component then starts looking for the required sequence of events again.

About Pattern parameters

When you build a Pattern component, you select the pattern type. The options are:

- **At least one event**
- **All of these events**
- **All of these events in sequence x times**

The **All of these events in sequence x times** pattern type assumes that in a sequence of events, it is likely that more than one occurrence of an event in the sequence may be initiated before the sequence has completed. Detect makes the assumption that the most recent occurrence of the event sequence is the one that is significant to firing the pattern.This assumption guarantees the most timely firing for patterns looking for events in sequence.

If you select **All of these events** or **All of these events in sequence x times**, you must set the following parameters.

- **Time Interval**—the unit of measurement
- **Quantity**— the meaning of this parameter varies depending on whether the Time Interval is for a floating or relative time frame

Together, these parameters define the time period during which the complete pattern must occur.

About floating time frames

Time intervals of **hour**, **day**, **week**, **month**, and **year** are used to create a floating time frame, where the occurrence of the first event in the pattern marks the start of the time frame. The complete pattern must occur within the selected number of time units.

For example, assume the time interval is **day** and the quantity is **3**. The first event of the pattern occurs. All remaining events must occur within 3 days in order to complete the pattern.

About relative time frames

Time intervals of **day of the week**, **day of the month**, **day of the quarter**, and **day** of the year are used to create a relative time frame. With a relative time frame, the Pattern component starts to watch for pattern events at a time determined by the selected quantity, and the complete pattern must occur within the selected time interval. With a relative time frame, the meaning of quantity is relative to the selected time interval, as illustrated in the following table.

Interval	Quantity	Meaning
day of the week	2	start Monday, running through Saturday (weeks start on Sunday)

Interval	Quantity	Meaning
day of the month	15	starting the 15th day of the month, running through the last day of the month
day of the quarter	1	starting the first day of the quarter, running through the last day of the quarter
day of the year	45	starting the 45th day of the year, running through the last day of the year

For example, assume the **Time Interval** is **day of the week** and **Quantity** is **3**. The pattern time frame is Tuesday through Monday. The first event of the pattern must occur no earlier than Tuesday. In order to complete the pattern, the remaining events must occur by the following Monday.

Adding a Pattern component

Add a new Pattern component as described in "Adding and editing components" on page 23. Specify the component properties as described in this section.

You use four tabs to specify a Pattern component.

- Components—The Components tab is where you select the event-generating components that are the basic building blocks of the pattern.
- Pattern parameters—The Pattern Parameters tab is where you set the pattern type.
- Properties—The Properties tab is where you specify the name and other information about the component. See "About component properties" on page 24.

To create a Pattern component

1. Select the Properties tab, complete the fields, and then click Set Parameters.

   An IF?TEN statement is started in the Summary area, with the current pattern's name listed under the THEN part of the statement.
2. Select the Pattern Parameters tab to set the pattern type, and then click **Add To Pattern**.
3. On the Components tab, select the component type to look for in building the pattern.

   An additional field is populated with existing components of that type.
4. Select the component.
5. Click **Add Event** to include the selected component in an IF/THEN statement in the Summary area of the Component Editor.

   If you want to re-start this Pattern component when the selected event occurs, click **Make Reset**.
6. Click **OK** to save the component.

Chapter 10. Containers and Container Manipulators

IBM Unica Detect provides the ability to create Container and Container Manipulator components. They are two different components, but they are related.

- Containers provide table-like storage and functionality.
- Container Manipulator components manipulate the data stored in containers.

About Container components

A Container component stores data (record sets) from transactions, tables, or other Containers for a specified length of time. Its purpose is to store the specified data for manipulation by other components.

You can define multiple containers. Each customer who generates transactional data has their own container(s).

You set parameters to define the following characteristics of a Container component.

- The type of data
- The maximum number of data elements
- The items to delete on an overflow condition
- The length of time to save the data
- The timing
- The method used to compress the items in stored in the container.

You can optionally define a Container to save only unique (distinct) pieces of data. For example, if your goal is to determine the different types of printer paper a customer purchases in one month, you would want to save the SKU of a paper purchase only once, even if the customer purchased it 3 times that month.

Types of data that a Container can hold

A Container component can hold any of the following.

- Numbers
- Dollar amounts
- Dates
- Text
- A record set with a structure defined by a Type Descriptor

Example of when you might use a Container

Suppose that your goal is to offer customers a discount for printer paper when they purchase over $200 in printer products in a month. You can define a container to store dollar amounts of printer purchases in the last month. Each time a customer purchases a printer product, the application will save the dollar amount of the product in a Container for that customer.

About duration in Containers

You control how a Container collects and stores data by specifying the duration. The types of duration settings are:

- floating
- relative

About floating duration in Containers

Use a Container with a floating duration you want to specify the length of time for which the Container stores a specific piece of data, based on the time when it was inserted into the Container.

For example, a Container is defined with a floating duration of 3 days. On Monday a piece of data is stored in the Container, and on Tuesday another piece of data is stored in the Container. On Thursday the piece of data that was stored on Monday will be deleted because is 3 days have passed. The piece of data that was stored on Tuesday will remain in the Container.

The floating duration is set as a specific number of time units, such as 2 days or 1 year. The available time intervals are minute, hour, day, week, month, year.

Note: Floating durations of 1 week and 7 days are identical. A floating duration of a month is not the same as 4 weeks because 4 weeks is a fixed 28 days, whereas a floating month goes by the date (such as the 23rd). Because months can vary in length, a floating month can vary in length.

About relative duration Containers

Use a Container with a relative duration when you want to specify the period during which the Container should collect data.

Relative duration is based on a fixed time in an hour, day, week, month, quarter, or year.

For example, suppose a relative duration is defined as starting on the 2nd day of the week, lasting for 2 days.

- On Sunday, a transaction occurs for which a piece of data is significant. However, the piece of data is not stored in the Container because the Container starts collecting data on Monday, the 2nd day of the week.
- On Monday, a transaction occurs for which a piece of data is significant, and it is stored in the Container.
- On Tuesday, a transaction occurs for which a piece of data is significant. It is stored in the Container.
- No more data is saved in the container until the following Monday or Tuesday.

When you create a Container with a relative duration, you must set the period start, time unit, time frame, and period length. Examples how relative duration Container settings define the Container are listed in the following table.

Period Start	Time Unit	Time Frame	Period Length	Resulting Characteristics
10	minute	hour	20	The container will start storing data on the 10th minute of the hour through 30th minute of the hour.

Period Start	Time Unit	Time Frame	Period Length	Resulting Characteristics
3	hour	day	4	The container will start storing data at 2am and continue through 5:59am.
2	day	week	2	The container will start storing data on Monday and continue through Tuesday until midnight.
2	week	month	1	The container will start storing data on the 2nd week of the month and continue through the end of that week.
2	month	quarter	2	The container will start storing data on the 2nd month the quarter and continue through the end of that quarter.
6	month	year	2	The container will start storing data on the June 1st and continue through the end of July.
1	day	year	32	The container will start storing data on the January 1st and continue until midnight February 1st.

About data compression in Containers

You can optionally use Data compression to save a summary of the information in a container when there is not enough room to hold every original data point. When compression is added to the container, compression then occurs automatically, reducing multiple data items within a container to a single data item, based on defined compression parameters and the compression fields in the Type Descriptor selected on the container details screen.

Note: When compression is used, the Type Descriptor that is used in the Container must be defined with compression parameters.

The compression process uses the field designated as the GroupBy to group multiple data items into a single data item, for those with the same GroupBy field value. The compression process then sums the values of the fields that are defined with the compression parameter.

The following table provides examples of the timing of compression and the results that occur.

Period Start	Time Unit	Period Length	Compression Results
0	Day	1	Compression occurs, immediately, during the run as the container is accessed. The data stored in the container that day is compressed. For example, on Monday, data is stored in the container. That data is compressed on Monday, at the completion of the transaction set, for a given ID.

Period Start	Time Unit	Period Length	Compression Results
1	Day	1	Compression occurs each day, during the run as the container is accessed. The data stored in the container the previous day is compressed. For example, on Monday, data is stored in the container. When the container is accessed during the Tuesday run, the data that was stored on Monday is compressed. Tuesday's data remains untouched.
7	Day	1	Compression occurs each day, during the run as the container is accessed. For example, if a user is making many transactions every week. By compression the information daily, you can for example determine the maximum, sum, or the average daily transaction for this customer over the course of a week.
2	Week	1	Compression occurs every two weeks, at the first run following a two-week cycle. The data stored in the container during the first week of the previous two-week cycle is compressed. Compression occurs as the container is accessed. For example, this month Sundays fall on the 1st, 8th, 15th, 22nd, and 29th. The last time the container was compressed was on the 1st. You are now doing a run for the 15th. The first time the container is accessed during the run, all items dated from the 1st through the 14th are compressed. The date - timestamp of record is the same as the uncompressed record that is closest to the 14th. Compression will occur when you do the run for the 29th. During this run the records date - timestamped from the 15th through the 28th are compressed.
1	Month	1	Compression occurs every month, at the first run following a one-month cycle. The data stored in the container during the previous month is compressed. Compression occurs as the container is accessed.

Adding a Container component

Add a new Container component as described in "Adding and editing components" on page 23. Specify the component properties as described in this section.

The Container editor screen has the following tabs.

- Details—The Details tab enables you to define the Container. See "To add a Container component" on page 51.
- Compression—The Compression tab enables you to add or remove compression for the Container. See "To add compression to a Container" on page 51.
- Properties—The Properties tab enables you to set a name and description for the Container, and select up to three Label values. The Properties tab is common to all component types. For a general description of all the properties tabs in the system, refer to "About component properties" on page 24.

To add a Container component

1. From the **Type** drop-down list on the **Details** tab, select the type of data the Container will store.

   If you want only unique values to be stored, click the **Distinct** checkbox.
2. In the **Maximum Value(s)** text box, enter the maximum number of rows of data the Container can store.

   Note: When a container reaches the maximum amount of data it can store, the overflow settings determine what happens when a request is made to store another piece of data.
3. Define the **On Overflow** parameters.

   When a maximum number of rows of data is reached, the overflow settings determine what happens when a request is made to store another piece of data.

   a. Select the **Delete** parameter.
      - **Minimum** - deletes the item containing the smallest value in the selected **Field**.
      - **Maximum** - deletes the item containing the largest value in the selected **Field**.

   b. Select the **Field** parameter. The drop-down lists the fields available in the selected Type.
4. Define the Duration parameters. See "About duration in Containers" on page 47.
5. Optionally, add compression as described in "About data compression in Containers" on page 49.
6. Click **OK** to save the component.

To add compression to a Container

You can optionally add compression to the Container. For details about compression, see "About data compression in Containers" on page 49.

Note: To use compression, the Type Descriptor that is used in the Container must be defined with compression parameters.

1. Select the **Period Start**.

   The **Period Start** parameter is a value from zero to ten that defines the starting point of the compression process. It is used in combination with the Time Unit parameter.
2. Select the **Time Unit**.

   The choices are Day, Week, Month, Quarter, and Year.

   The **Time Unit** parameter is used in combination with the Period Start parameter to define the starting point of compression. And, it is used in combination with the Period Length parameter to define the quantity of data that will be compressed.
3. Select the **Period Length**.

   The **Period Length** parameter is a value from one to ten that defines the starting point of the compression process. It is used in combination with the Time Unit parameter.
4. Continue to the next section to set the properties for the container.

About Container Manipulator components

Container Manipulators are components that perform operations on the data stored in containers. When a specified event occurs and any defined conditions are met, a Container Manipulator performs its defined operations on a specified Container component.

When a Container Manipulator is triggered, it can notify other components that it has changed the contents of a container.

Events that a Container Manipulator can recognize

Container Manipulator components can recognize events triggered by any of the Detect components that can trigger an event.

- Simple Event
- Backward Looking Inactivity
- Forward Looking Inactivity
- Another Container Manipulator
- Pattern
- Action

Container Manipulator components can also recognize artificial transactions and events triggered by an Action component. An artificial transaction is a system-generated event, created during run-time, that is based on an optional parameter of the engine process. Examples of artificial transactions are: End of Run End of Day Transaction.

Sources of Container Manipulator data

The Container Manipulator can act on data that comes from the following sources:

- field from the transaction that initiated the evaluation, for example the transaction that triggered a Simple Event
- field from a profile data source
- field value from a Select Function
- value calculated in a Math component, for example 20% of the year-to-date amount a customer has purchased
- result of a Select Function
- defined constant
- operations performed on contents of another Container, including the results of a function. See “About functions” on page 27 for a description of functions.

Example of how to use a Container Manipulator

Your goal is to offer a discount to customers who purchase more than $500. To do this you would create a trigger system by doing the following:

- Define a Container, Purchase Amounts, to store the value of purchases. Allow a maximum of 100 purchases.
- Create a Simple Event component, Purchase Transactions, that triggers when a customer makes a purchase transaction.
- Create a Container Manipulator component, Save Purchase Amounts, that looks for the event triggered by Purchase Transaction and places the dollar amount of the purchase in the Purchase Amounts Container.

- Create a second Container Manipulator component, Purchases Greater Than 500, that looks for the occurrence of Save Purchase Amounts. It would evaluate the sum of the amounts saved in the Purchase Amounts Container and when it is greater than $500, clears the contents of the container and triggers an event.

The event triggered by Purchases Greater Than 500 would be recognized by an Action component that creates a message in the outcome table.

About conditions in Container Manipulators

Optionally, you can include one or more conditional expressions in a Container Manipulator. A conditional expression lets you to refine the conditions under which the data in a Container is manipulated, and under which the Container Manipulator component triggers.

The conditional expression is a comparison clause (or combination of comparison clauses) that evaluates specific data. It can be built as a combination of comparison clauses, where the clauses are joined by either AND or OR. For example, you can express "If (A and (B or C)".

Sources of conditional expression data

The data in a conditional expression used in a Container Manipulator may come from any of the following sources.

- data source
- the result of a Math expression
- data (or manipulated data) in a Container
- a Lookup Table
- a result of a Select function
- a constant (with another specific piece of data)

About logical and comparison operators

See the following for information about operators in comparison expressions.

- "About logical operators" on page 29
- "About comparison operators" on page 30

Adding a Container Manipulator component

Add a new Container Manipulator component as described in "Adding and editing components" on page 23. Specify the component properties as described in this section.

You use two tabs to specify a Container Manipulator component.

- Details—The Details tab is where you build the Container manipulator by doing the following.
  - selecting the event
  - optionally, building a conditional expression (the Optional If clause)
  - defining the manipulation parameters (the Do clause)
- Properties—The Properties tab is where you specify the name and other information about the component. See "About component properties" on page 24.

To create a Container Manipulator component

1. On the Details tab, expand the **ON** section.
2. Select a component type from the **Event** drop-down list.
3. Select the name of the component from the second drop-down list.
4. Click **Set Event**.
5. Optionally, expand the **Optional IF** section to add a conditional expression.
   See "About conditions in Container Manipulators" on page 53.
6. Add the manipulation parameters as defined in the following procedures.
   - "To define the manipulation parameters using Insert or Soft Insert."
   - "To define the manipulation parameters using Delete" on page 55.
7. Click **OK** to save the component.

To define the manipulation parameters using Insert or Soft Insert

Repeat this procedure for each Insert or Soft Insert parameter you want to add.

1. On the Details tab, expand the **DO** section and select one of the following:
   - **Insert** insert the data, defined by the **Value** parameters, into the container, defined by the selection chosen from the "Into Container" parameter. The **Value** can be the value of a field in a data source, the value of a field in a Lookup table, the value of a result calculated by a Math expression, the value of a field in a Container, the value of a field as a result of a Select Function, or a constant value.
   - **Soft Insert** - does the same function as Insert, but waits for the transaction being processed to be completed before the value is inserted into the Container.
2. Select **Value Type**.

   When choosing Container, Select Function, or Lookup table you must select one of the Value Functions. See "About functions" on page 27.
3. Map the selected **Insert** value to the selected **Into Container** definition. This process assigns a field from one of the permitted sources into a field in the container.
4. Choose a field from the **Container Definition** tree.
5. Click the associated parent-child button to expand the field definition.

   Note: The data type of the field value being inserted must match the data type of the field in the container definition.
6. Enter the **Index** number of the chosen container definition field in the **Map to Index** text box.
7. Click the **Map to Index** button.
8. Repeat the mapping steps for the desired number of fields.
9. Select the container in which to place the results from the **Into Container** dropdown.
10. Click **Add Task** to add the **DO** action to the right side of the window.
11. Click **Save**.

To define the manipulation parameters using Delete

Repeat this procedure for each Delete parameter you want to add.

1. On the Details tab, expand the DO section and select **Delete** in the **Do** drop-down list.
2. Select the **From Container** from the drop-down list of containers.
3. Select a **Where Field** from the drop-down list of fields in the selected container.
4. Select a comparison **Operator** to build the clause. The choices are equal, not equal, greater than, greater than or equal, less than, and less than or equal.
5. Select a **Value Type** and **Value**.
   - For **Math**, select a math expression from the list presented in the "Value" drop-down.
   - For **Literal**, enter a constant in the "Value" text box.

   Note: Any row in the container that has a value that meets the criteria defined by the **Criterion Expressions** parameters will be deleted.
6. Click the **Add Criterion** button to use your selections to start building the Criterion Expression.

   Note: The criterion will be added using AND.
7. Click **Save**.

Chapter 11. Select Function Components

The Select Function component provides a means of filtering the information in a table according to defined criteria. It is comparable to the SQL SELECT statement.

The Select Function returns a record-set object whose type and format are determined by a previously defined Type Descriptor. If the Type Descriptor does not exist, it must be created using the Type Descriptor Editor (described in the *IBM Unica Detect Administration Guide*) before you can create a Select Function.

Examples of where to use a Select Function

You can use Select Functions wherever a record-set object is required. For example:

- As the data source of another Select Function
- As a recordset data source of a Join Function
- As a value that is inserted into a container via the Container Manipulator component
- In the Outcome tab of the Action Component Editor
- As an operand in a Math Expression component
- As an operand in a Qualifier component
- In the IF section of a Container Manipulator component

Sources of Select Function data

A Select Function uses a single originating data source. The data source can be any of the following.

- Container component
- Join Function component
- Lookup table
- Select Function component
- Data source

Methods for filtering Select Function data

You can specify the criteria for reducing the information presented by the originating source using any of several different methods.

- Selectively mapping fields from the originating source to the output
- Mapping all fields from the originating source to the output
- Defining a single WHERE clause or multiple WHERE clauses

As you map each field, you can add additional filtering criteria.

- Results of a function. See “About functions” on page 27.
- Data–includes all the values in the selected field
- Group By–uses one field as a grouping field
- Order By–uses one field to determine the ordering of the output results

Detect uses a WHERE clause to evaluate a field value from the originating source. If the field value meets the specified criteria, the originating source record is included in the output, as determined by the selected field mapping. Multiple WHERE clauses can be used

About Select Functions with Join Function data sources

A Select Function component that uses a Join Function as its data source can be processing- intensive because it is executed at run time on what can be a different set of records for each entity.

However, a Select Function can be used with the Is Member Of option in a Qualifier component. When Is Member Of is used, the selection is done once as the rules are loaded, and it is therefore much more efficient. For added efficiency, you can use the Is Member Of option to refer to records in a lookup table, which by definition are the same for all entities. Using a Select Function in this way is a method for creating table-driven rules that are easy to manipulate and modify.

For more about Is Member Of feature, see "About the Is Member Of comparison operator" on page 30

Adding a Select Function component

Add a new Select Function component as described in "Adding and editing components" on page 23. Specify the component properties as described in this section.

The Select Function editor has two tabs:

- Details—The Details tab is where you specify the Select Function component criteria.
- Properties—The Properties tab is where you specify the name and other information about the component. See "About component properties" on page 24.

To specify a Select Function

1. On the **Details** tab, select a data source from the **Source** drop-down lists.
2. Choose an action from the **Task** drop-down list.
3. Select from the **Field List**.

   The SQL builds on the right side of the window.
4. The action you chose in step 2 will determine what options you see. For example:
   a. If you chose **Group By** or **Order By**, use the buttons to add or remove Group/Order settings.
   b. If you chose **Select**, do the following select a field, and optionally, select a function.
      - Select a field.
      - Optionally, select a function from the drop-down list. For more information, see "About functions" on page 27.
      - Enter the Index number of the chosen output field in the **Map to Index** text box, then click **Map to Index**.
      - Select the **Output Type Descriptor** from the drop-down list.

The data type of the **Source** field must match the data type of the **Output Type Descriptor**.

The associated field tree displays below the selected type.

To expand a field's information, click the associated parent-child button.

5. Define the **WHERE Clause** criteria. From the dropdown lists, select the following parameters for the first operand.
   a. The field you want to select on
   b. The operator to use.
   c. The source type
   d. The source name
   e. The source field
6. Click **Add Clause**.

   You can continue to build on the WHERE clause. Additional WHERE Clauses are joined by AND.
7. Click **OK** to save the component.

Chapter 12. Backward and Forward Looking Inactivity Components

Backward Looking Inactivity (BLI) components are used when, as the result of some present event, you want to verify that some other event did not occur within a specified previous time frame. For example, when a customer withdraws $10,000 from his account, you might want to test to see that she has not made any other withdrawals over $5,000 within the past two months.

The BLI component requires the following three parameters.

- Terminating Event —the event that initiates the Backward Looking Inactivity
- Inactivity Time Interval —the period of time between the occurrence of the Terminating Event and the non-occurrence of the Tracked Event
- Tracked Event —the inactivity event, which may be the triggering of any of the following components.
  - Simple Event
  - Pattern
  - Forward Looking Inactivity
  - Another Backward Looking Inactivity
  - Container Manipulator
  - Action

The occurrence of the Terminating Event arms the BLI component and sets the end date for the Inactivity Time Interval. If the Tracked Event did not occur in the past time interval defined, the Backward Looking Inactivity will fire.

The Tracked Event has relevance only when the Terminating Event triggers and arms the BLI component.

Examples of Backward Looking Inactivity Components

This section provides examples of BLI components.

Example 1 of a BLI component

A BLI component named Customer Retention Successful might look for customers who make a store purchase after making no online purchases for 6 months. These are the three parameters of this Backward Looking Inactivity component:

- Terminating Event—customer makes a store purchase
- Tracked Event—online purchase
- Inactivity Time Interval—prior 6 months

When a customer makes a store purchase, this component is evaluated and the system looks over the Inactivity Time Interval of prior 6 months to see if the Tracked Event online purchase has occurred. If an online purchase has not occurred in the past six months, the component triggers.

Example 2 of a BLI component

A BLI component named Customer Retention Successful might look for customers who make a major deposit and them make no other deposits or withdrawals for one month or more. These are the three parameters of this Backward Looking Inactivity component:

- Terminating Event—make a major deposit
- Tracked Event—make no other deposits or withdrawals
- Inactivity Time Interval—1 month or more

Adding a BLI component

Add a new BLI component as described in "Adding and editing components" on page 23. Specify the component properties as described in the following procedure.

You use the following two tabs to build a BLI component.

- Properties—The Properties tab is where you specify the name and other information about the component. See "About component properties" on page 24.
- Components—The Components tab is where you specify the BLI parameters. For the basic procedure, see "To specify a BLI component."

  The Components tab is divided into the following areas.
  - Inactivity Time Interval defines the length of time prior to the Terminating Event, during which a check is made for the Tracked Event.
  - Tracked Event defines the inactivity event, a specific Simple Event, Pattern, Forward Looking Inactivity, Backward Looking Inactivity, or Container Manipulator.
  - Terminating Event defines the event that initiates the Backward Looking Inactivity.
  - Behavior defines how the Backward Looking Inactivity acts when a Tracked Event has not occurred.

To specify a BLI component

If you need to refresh the drop-down lists while you are creating the BL:I component, click **Refresh Events**.

1. In the Inactivity Time Interval area, select an option in the **Time Quantity** drop-down list.

   The other field selections in this section adjust depending on what option you choose.
   - If you chose **Literal Value**, then select a number from 1 through 100.
   - If you chose **Data Source Field**, then select a data source, and a datasource field.
   - If you chose **Math Expression**, then select an existing Math component.
2. In the Inactivity Time Interval area, select a **Time Unit** (Hour, Day, Week, Month, or Year).
3. In the Tracked Event area, select an **Event Type** and select an **Event Name** from the drop-down list of existing, corresponding components.
4. In the Terminating Event area, select an **Event Type** and select an **Event Name** from the drop-down list of existing, corresponding components.

5. If you want to reset the date of the last occurrence of the Tracked Event to the current date when the Backward Looking Inactivity fires, then check the **Treat firing of this Backward Inactivity as a Tracked Event** checkbox. If this field is not checked, the date of the last occurrence of the Tracked Event remains unchanged.
6. In the Behavior area, select an option in the **Firing Behavior without a Tracked Event** drop-down list.

   This parameter defines when the inactivity fires when the engine recognizes the Termination Event for customers where there is no Tracked Event. The options are as follows.

   - **Fire Immediately**—the inactivity fires as soon as it recognizes the Terminating Event.
   - **Consider Creation Dates**—the creation date of the Backward Looking Inactivity and the date the UserId was created in Visitor History will be used to determine whether the inactivity should fire.

     The system considers the creation date of both visitor history record and the creation date of the rule. For example, if a rule was created three weeks ago and the user is new as of yesterday, there may not be enough history on the user for the trigger to fire.
7. Click **Set Parameters** to add the selected events and Inactivity Time Interval to the BLI.
8. Click **OK** to save the component.

About Forward Looking Inactivity components

Forward Looking Inactivity (FLI) components look for an event, then look for a second event that would normally occur after a specified period of time.

If the first event occurs, a time period is imposed. If the second event does not occur during the imposed time period, that is a break in the normal pattern, and the FLI fires.

The FLI event types can be any of the following event-generating components.

- Simple Event
- Pattern
- Another FLI
- BLI
- Container Manipulator

The FLI component requires the following three parameters.

- Initial Event —the event that initiates the FLI component
- Inactivity Time Interval —the period of time between the occurrence of the Initial Event and the non-occurrence of the Tracked Event
- Tracked Event —the inactivity event

Examples of Forward Looking Inactivity components

This section provides examples of FLI components.

Example 1 of an FLI component

An FLI named No additional purchases in 1 month has the following parameters.

- Initial event—customer makes a purchase
- Tracked event—customer makes another purchase
- Inactivity time interval—month following the first purchase

On June 10, a customer makes a purchase.The FLI starts monitoring, watching for that customer to make another purchase. If no other purchase occurs by July 10, the FLI triggers for that customer.

Example 2 of an FLI component

An FLI named No Deposits in 1 month has the following parameters.

- Initial Event—make a deposit
- Tracked Event—makes no other deposits
- Inactivity time interval—within the next 1 month (which is From time of Initial Event, where the Time Quantity is a literal value)

On June 10, a customer makes a deposit.The FLI starts monitoring, waiting for that customer to make another deposit. If none occur by July 10, the FLI triggers for Mr. Smith.

With a variable "Time Quantity" the time interval may be different for each customer.

Example 3 of an FLI component

An FLI named No Deposits in a variable number of months has the following parameters.

- Initial Event—make a deposit
- Tracked Event—makes no other deposits
- Inactivity time interval—within a variable number of months (which is From time of Initial Event, where the Time Quantity is based on a data source field)

On June 10, a customer makes a deposit. The FLI starts monitoring. When the value in the Time Quantity data source field equals 2, the FLI triggers for that customer if another deposit does not occur by August 10. When the value equals 3, the FLI would wait until September 10.

With an inactivity interval of Until a Specified Time, the occurrence of the Initial Event tags the customer as one to monitor for inactivity until a defined day, week, month, quarter, or year. This anchors the actual monitoring for the inactivity to a fixed calendar schedule.

Example 4 of an FLI component

An FLI named No Deposits in the month has the following parameters.

- Initial Event—make a deposit
- Tracked Event—makes no other deposits
- Inactivity time interval—until the 1st day of the month (which relates to Until a Specified Time)

On June 10, a customer makes a deposit. The FLI starts monitoring. If another deposit does not occur by July 1, the FLI triggers for Mr. Smith.

Adding an FLI component

Add a new Forward Looking Inactivity component as described in "Adding and editing components" on page 23. Specify the component properties as described in the following procedure.

You use the following two tabs to build an FLI component.

- Properties—The Properties tab is where you specify the name and other information about the component. See "About component properties" on page 24.
- Components—The Components tab is where you specify the BLI parameters. For the basic procedure, see "To specify an FLI component." For additional information, see the remainder of this chapter.

To specify an FLI component

1. Set the **Initial Event**.
   a. Select an **Event Type** (forward inactivity, backward inactivity, simple event, pattern, or Container Manipulator).
   b. Select an **Event Name** from the drop-down list of existing, corresponding components.
2. In the **Inactivity Time Interval** area, choose the **Type** of interval.
   - If you choose **From time of Initial Event**, the other field selections in this section adjust depending on what you choose for **Time Quantity**.
     – If you chose **Literal Value**, then select a number from 1 through 368 and a **Time Unit** (Hour, Day, Week, Month, or Year).
     – If you chose **Data Source Field**, then select a data source, a datasource field, and a **Time Unit** (Hour, Day, Week, Month, or Year).
     – If you chose **Math Expression**, then select an existing Math component and a **Time Unit** (Hour, Day, Week, Month, or Year).
   - If you chose **Until a Specified Time**, the system retrieves information on an individual basis and makes the FLI calculation based upon values from a data source field, a math expression, a literal value, or a time constant, depending on your selection. The options vary, depending on which option you select for the **Until** field.
     – If you chose **Literal Value**, then select a number and a time unit from the dropdown lists. In the **of** dropdown, choose the time frame. (The time choices you see are dependant on the time unit setting.)
     – If you chose **Data Source Field**, then select a data source, a datasource field, and a **Time Unit** (Hour, Day, Week, Month, or Year).
     – If you chose **Math Expression**, then select an existing Math component and a **Time Unit** (Hour, Day, Week, Month, Quarter). In the **of** dropdown, choose the time frame. (The time choices you see are dependant on the Time Unit setting.)
     – If you chose **Time Constant**, then select a time unit from the dropdown lists. In the **End of** dropdown, choose the time frame. (Day, Week, Month, Quarter, or Year).

Note: The selected unit of time value defines a measurable unit of time occurring within the time frame and must therefore be smaller than the selected time frame.

3. Optionally, fill the **Tracked Event** section.
   a. Check the **Look for Inactivity of Tracked Event** box.
   b. Select an **Event Type** (backward inactivity, simple event, pattern, or Container Manipulator).
   c. Select an **Event Name** from the drop-down list of existing, corresponding components.
4. Click **Set Parameters** to add the selected events and Inactivity Time Interval to the FLI.
5. Click **OK** to save the component.

About the reset options in the Intial Event parameter

The Initial Event is the event that initiates the Forward Looking Inactivity. You set this parameter in the Initial Event section of the FLI Component Editor. A variable checkbox in this section provides several reset options. This checkbox appears only if the **Look for Inactivity of Tracked Event** checkbox is selected in the Tracked Event section.

If the Initial Event and the Tracked Event are the same, the checkbox is automatically selected and cannot be cleared.

When the Inactivity Time Interval type is set to **From Time of Initial Event**, the checkbox is labeled **Each occurrence reinitializes**.

- If the checkbox is selected, each occurrence of the Initial Event resets the start of the Inactivity Time Interval to the date of the most recent occurrence.
- If the checkbox is left clear, the first occurrence of the Initial Event sets the time span, and subsequent occurrences of the Initial Event during the defined time interval does not change the time for which the forward looking inactivity is set.

When the Inactivity Time Interval type is set to **Until a Specified Time**, the checkbox ix labeled **Set to the very next calendar period if the FLI is already set**.

- If the checkbox is selected, the occurrence of an Initial event when the FLI is already set causes the forward looking inactivity to be reset from the end of the current calendar period to the end of the very next calendar time period.
- If the checkbox is left clear, the first occurrence of the Initial Event sets the time span, and subsequent occurrences of the Initial Event during the defined time interval do not change the time for which the forward looking inactivity is set.

About Inactivity Time Interval options

The Inactivity Time Interval is the period of time that the FLI monitors after the Initial Event. The Inactivity Time Interval can be defined to track data for a specified amount of time, or until a specified event occurs or does not occur within a specified time.

When defined **From time of Initial Event**, you have the following options.

- Directly specify the length of time by choosing a literal time quantity.
- Specify a variable time based on the value in a data source field or the result of a math expression.

When defined **Until a Specified Time interval**, you have the following options.

- Either the time interval expires without the occurrence of the Tracked Event, and the FLI fires, or
- the Tracked Event occurs, interrupting the time interval, and the FLI does not fire.

About the Tracked Event FLI parameter

The Tracked Event is the inactivity event. This setting is optional.

If you want the FLI to trigger a specified time after the occurrence of the Initial Event without watching for the non-occurrence of another event, do not select a Tracked Event. In that case, the FLI acts as a timer. For example, the FLI might arm when a customer purchases a printer. It then waits 1 month and writes an outcome that results in offering a discount on an ink cartridge.

When a Tracked Event is included in the FLI, the occurrence of the Initial Event arms the component and sets the end date for the Inactivity Time Interval.

- If the Tracked Event does not occur in the future time interval defined or no Tracked Event is defined, the Forward Looking Inactivity fires.
- If the Tracked Event does occur, then it interrupts the time interval and the component does not fire.

Chapter 13. Join Functions

The Join Function component joins two record-set sources. A common field from each originating source is used to connect the two sources.

At the most, the returned record-set object may contain the union of all the fields from both incoming record-set sources. At the least, the returned record-set object may contain as little as one field from either of the two incoming record-set sources.

The Join Function component returns a record-set object whose type whose type and format are determined by a previously defined Type Descriptor. If the Type Descriptor does not exist,

it must be created using the Type Descriptor Editor (described in the *IBM Unica Detect Administration Guide*) before you can create a Select Function.

Uses for the Join Function component

You can use the Join Function component wherever a record-set object is required, as follows.

- As a Recordset Source of another Join Function
- As the Source of a Select Function
- As a value that is inserted into a Container component via the Container Manipulator component
- As the subject of a record-set operator, which can be used as follows.
  - In the Outcome tab of the Action editor
  - As an operand in the Math Expression editor
  - As an operand in the Qualifier editor
  - As an operand in the Advanced Details section of the Simple Event editor

Sources of Join Function component data

A Join Function uses two originating sources. The sources can be any combination of a the following.

- Container component
- Join Function
- Lookup Table
- Select Function
- Transaction data source

The originating sources are referred to as **Recordset Source (Left)** and **Recordset Source (Right)**.

About the available join types

Two join types are available:

- Inner Join—An Inner Join requires that one field from both **Recordset Source (Left)** and **Recordset Source (Right)** be selected to connect the two originating sources.
- Minus–Rows in **Recordset Source (Left)** that are not in **Recordset Source (Right)**, where the data type definition for both are the same

About Join Function output

Fields from the originating sources can be mapped to the output:

- Selectively, mapping each field one-at-a-time
- Using the **Map All** button, mapping all fields from **Recordset Source (Left)** followed by all fields from **Recordset Source (Right)**

The data type of a field from an originating source must match the data type of the field it is mapped to in the output. The name of the field in the output may be different.

Adding a Join Function component

Add a new Join Function component as described in "Adding and editing components" on page 23. Specify the component properties as described in this section.

The Join Function uses a pre-defined Type Descriptor for output. If the Type does not exist, first create it using the Type Descriptor Editor and then create the Join Function.

The Select Function editor has two tabs:

- Details—The Details tab is where you specify the Join Function component criteria.
- Properties—The Properties tab is where you specify the name and other information about the component. See "About component properties" on page 24.

To specify a Join Function component

1. In the **Details** tab, select a **Join Type** from the drop-down list. The options are:
   - **Inner Join**. Matching rows only
   - **Minus**. Rows in **Recordset Source (Left)** that are not in **Recordset Source (Right)**, where the data type definition for both are the same
2. Select the **Recordset Source (Left)** source type and actual source from the drop-down lists.

   The **Field List** displays the fields defined by the specified **Recordset Source (Left)** parameters.
3. Select the **Recordset Source (Right)** source type and actual source from the drop-down lists.

   The **Field List** displays the fields defined by the specified **Recordset Source (Right)** parameters.
4. Select one field from the **Field List** for the **Recordset Source (Left)** and one field from the **Field List** for the **Recordset Source (Right)** on which to join.
5. Click the **Connect Selected Fields** button to create the join.

   The join information appears on the right of the window.

Note: A Minus join type does not require a a field on which to join. The join operation is performed on sources left and right in their entireties.

6. Select the **Output Type Descriptor** from the drop-down list.

   The associated field tree displays below the selected type.

7. Map fields from the originating Recordset Sources, Field Lists, to the Output Type Descriptor field tree.
   a. In the **Field List**, click the source field to be mapped.
   b. Choose a field from the output tree.
   c. Click the associated parent-child button to expand the field definition.

      Note: The data type of the source field must match the data type of the output field.
   d. Enter the Index number of the chosen output field in the **Map to Index** text box.
   e. You can find the index number of the field by expanding the field details in the field tree (by clicking on the +).
   f. Click the **Map to Index** button.

      Note: The editor maps the side last clicked on, even though both sides maintain a highlighted selection.
   g. Repeat the mapping steps for the desired number of fields.

      Note: The **Map All** button will map all the fields from the Recordset Source (Left) followed by all the fields from the Recordset Source (Right) to the selected output.

8. Click **OK** to save the component.

Chapter 14. Trend Components

All trend components (whether trend, spike, or exceeded standard deviations) are used as filters for Action, Container Manipulator, or Qualifier components. They refine the triggering of those components to just those customers characterized by the parameters defined in the component. For example, if a trend of upward usage is attached to an Action, then the Action fires only if the upward usage is true.

The Trend component is similar to the qualifier component in that it represents a true or false value. A Trend component evaluates to true when the referenced data values satisfy all the logical conditions defined in the component. Trend components never fire on their own. They are used internally by other components.

Note: In cases where the user interface refers to a trend component firing, it means that the trend resolves to true. Trend components (whether type trend, spike or exceeded standard deviation) cannot truly fire on their own the way other components can.

Sources of Trend component data

A trend can track any single numeric value in a container or select function. The trend can also track increasing or decreasing density (counts) of non-numeric values; but the use cases for this are rare.

A Trend component can reference data from the following components.

- Container
- Select Function

Types of trends the Trend component can track

The Trend component can track the following trend types.

- **Trend**—detects basic trends. For example, the average rolling monthly balance has been increasing or decreasing by 10% over the last three months.
- **Spike**—detects a change in activity. For example, a deposit is 50% larger or smaller than any other deposit made in the last 3 months.
- **Exceeded Standard Deviation (ESD)**—detects activity outside of the standard variance for that particular entity (such as customer or account) for a specific time period. For example, a monthly balance that is greater than or less than the average monthly balance for the past 12 months by two standard deviations.

Note: For simplicity, this document refers to Trend components by their type: trend (for basic trend), spike, and exceeded standard deviation (ESD).

Related topics:

- "About the trend type of the Trend component" on page 79
- "About the spike type of the Trend component" on page 83
- "About the spike type of the Trend component" on page 83

About subgroups

Use subgroups within a Trend component to further refine the way data is evaluated. Groupings allow you to specify smaller units of time within the component to enhance the accuracy of to component results.

Combinations of function and data type are allowed

The user interface limits the combinations of function and data type that you can create to only those that are valid. For example, you cannot create a combination of a mathematical function performed on a string or date field.

The following table shows, for each data type, the valid combinations of period and subgroup functions. For example, if data type is Date and there are no subgroupings, then Count and Count Distinct are the only valid group function. In addition, if Count or Count Distinct are the subgroup function, then any period function is valid.

Data type	Subgroup function	Period function
Integer	(any)	(any)
Double	(any)	(any)
Date	(none)	Count, Count Distinct
	Count, Count Distinct	(any>
	First, Last, Maximum, Minimum	Count, Count Distinct
String	(none)	Count, Count Distinct
	Count, Count Distinct	(any)
	First, Last, Maximum, Minimum	Count, Count Distinct

Related tasks:

"About the trend type of the Trend component" on page 79

"About the spike type of the Trend component" on page 83

"About the exceeded standard deviations type of the Trend component" on page 86

About time boundaries for trends

The trend components provide the ability to sub-divide the elements in a container or a select function into time periods. Some of these time periods have a natural definition (such as calendar day) and some do not (such as rolling month).

There are two general types of periods: calendar and rolling.

- Rolling periods start at the timestamp on the day of the transaction and go back in time to that same timestamp of on an earlier day.
- Calender periods (calendar days, calendar weeks, and calendar months) always begin and end at midnight.

When you define a trend component, you select calendar or rolling for only the last time period, and Detect adjusts all previous time periods accordingly. For

example, if a spike period is rolling, then the historical period will also be rolling. And if the spike period is calendar, then the historical period will also be calendar.

Note that some subgroups are disallowed because they result in an irregular division of a time period. For example, you are not allowed to subgroup a monthly period into weeks, because not all months cannot be evenly divided into an even number of weeks. If you want to approximate this monthly subgrouping, pick periods that are 4 weeks long and then divide those into weekly subgroups.

About rolling bounded periods

Under certain conditions, the system imposes special rules to define boundaries for rolling periods. These periods are referred to as rolling bounded periods. Rolling bounded periods are calendar weekly or monthly periods that do not necessarily start on the first day of the week or the first day of the month. Rolling bounded periods are required to tight-fit historical periods to a spike period of a calendar day or in some cases a calendar week. (The spike period is used in spike Trends and in exceeded standard deviations Trends.)

Time boundaries for trends

The following table describes the time boundaries that can be used in trends, and explains the rules that define the boundaries of the trend time periods. All examples are shown with granularity of 1 second.

Note: The rolling bounded time boundaries do not appear as settings in the user interface. However, they are included here to explain how the time boundaries behave when the system requires them to tight-fit historical periods to a spike period of a calendar day or in some cases a calendar week.

Time Boundary	Description
Calendar Day	Includes midnight of the transaction day up to but not including midnight of the next day. For example: If the current transaction is dated 2009-03-17 09:22:00, then the boundaries of its calendar day are 2009-03-17 00:00:00 – 2009-03-17 23:59:59 inclusive
Rolling Day	Includes everything greater than this second of the previous day up to and including the second of this current transaction. If the current transaction is dated 2009-03-17 09:22:00, then the boundaries of its rolling day are 2009-03-16 09:22:01 – 2009-03-17 09:22:00 inclusive
Calendar Week	Includes Sunday midnight up to but not including midnight of the following Sunday. For example: If the current transaction is dated (Tuesday) 2009-03-17 09:22:00, then the boundaries of its calendar week are (Sunday) 2009-03-15 00:00:00 – (Saturday) 2009-03-21 23:59:59 inclusive
Rolling Week	Includes everything greater than this second of the previous week up to and including the second of this current transaction. If the current transaction is dated (Tuesday) 2009-03-17 09:22:00, then the boundaries of its rolling week are (Tuesday) 2009-03-10 09:22:01 – (Tuesday) 2009-03-17 09:22:00 inclusive

Time Boundary	Description
Calendar Month	Includes midnight of the first of the month up to but not including midnight of the first of the next month. Note that calendar months vary in length. For example: If the current transaction is dated 2009-03-17 09:22:00, then the boundaries of its calendar month are 2009-03-01 00:00:00 – 2009-03-31 23:59:59 inclusive
Rolling Month	Includes everything greater than this second of this day of the month of the previous month up to and including the exact second of this current transaction. For example: If the current transaction is dated 2009-03-17 09:22:00, then the boundaries of its rolling month are 2009-02-17 09:22:01 – 2009-03-17 09:22:00
Rolling Bounded Week*	Used for historical periods when the spike period is of type calendar. For a particular day of the week, includes midnight of that day of the week up to but not including midnight of that same day of the next week. For example: If the current transaction is dated 2009-03-17 09:22:00, then the boundaries of its calendar day are 2009-03-17 00:00:00 – 2009-03-17 23:59:59. Also, the boundaries of the rolling bounded week that precedes the calendar day are: 2009-03-10 00:00:00 – 2009-03-16 23:59:59
Rolling Bounded Month*	Used for historical periods when the spike period is of type calendar. For a particular day of the month, includes midnight of that day of the month up to but not including midnight of that same day of the next month For example: If the current transaction is dated 2009-03-17 09:22:00, then the boundaries of its calendar day are 2009-03-17 00:00:00 – 2009-03-17 23:59:59. Also, the boundaries of the rolling bounded month that precedes the calendar day are: 2009-02-17 00:00:00 – 2009-03-16 23:59:59

Details about month boundaries

This section provides details about the logic used to manage boundaries related to months. Months require careful logic because their length can vary (between 28 and 31 days).

End of month boundaries: end date 1 through 28

If the end date is 1 through 28, then the end dates of each previous month will be that same corresponding day of the month.

For example if the end date is 3/28, then the end dates of previous months will be 2/28, 1/28, 12/28, 11/28, 10/28, etc.

End of month boundaries: end date 29 through 31

If the end date is 29 through 31, then the end dates of the each previous month will be the same corresponding day of the month unless a month is short and lacks that day of the month. In that case the end date for the short month will be the last calendar day of the month in that month.

If a month end date reverts to the last day because the month is short of days, then the month end of the month previous to the shortened month will recover the

lost end days. This behavior holds true in the trend within monthly subgroups of rolling monthly periods as well as in trends of a series of months.

For example, if the end date is 3/31, then the end dates of previous months will be 2/28, 1/31, 12/31, 11/30, 10/31, etc.

Note: These monthly end dates are preserved even if they are embedded in a trend period that ends on 30. For example, if the trend has four three-month time periods in it, and the end date of the last period is 3/31, then going backward the end dates of the four trend periods will be 3/31, 12/31, 9/30, and 6/30. If these periods are subdivided into monthly subgroupings, then the one ending 9/30 will have three monthly subgroups ending 7/31, 8/31, and 9/30. Note that the July and August subgroups end on 31 even though they belong to a period that ends on 30. The reason for this is that the end date of the subgroups is governed by the 3/31 end date of the last trend period and not the 9/30 end date of the period.

Beginning points, ending points, and end-of-month arithmetic

Every period has a beginning and end point.

For monthly rolling periods, the beginning and end points are calculated as follows.

- The end point is the date and timestamp of the transaction.
- The beginning point is the previous month with the same day of the month and timestamp as the current transaction.

In contrast, for monthly calendar periods, the beginning and end points are calculated as follows.

- The beginning point is midnight of day x to midnight of day x of the next month.
- To ensure that contiguous time periods are continuous and do not overlap, one of the end points is included in the calculation of the period and the other will not.

With calendar or rolling bounded periods, the midnight of the beginning point is included but the midnight of the ending point is not.

For example: For June, the calendar month includes midnight of June 1 to midnight of July 1. The midnight of June 1 is included in the time period but the midnight of July 1 is not.

With rolling periods, the end point of the transaction date is included, but the beginning point of the same day of the month and the same time of the previous month is not.

For example: For a transaction occurring on June 29 at 23:59:59, the end point of June 29 23:59:59 is included, but the beginning point of May 29 at 23:59:59 is not.

Note: End of month boundaries are calculated moving backward using the end point of the last period as the reference point. So if the end point of a period is 6/30, then the end point of the previous month is 5/30. And if the end point of a period is 7/1, then the end point of the previous month is 6/1.

Around the end of a month, a 1 second difference in two transactions could mean a difference of a day or more in the monthly duration.

For example: A rolling month period that ends on June 29 at 23:59:59 spans from May 29 at 23:59:59 (point not included) to June 29 at 23:59:59 (point included). And a rolling month period that ends on July 1 at 00:00:00 spans from June 1 at 00:00:00 (point not included) to July 1 at 00:00:00 (point included). Here a difference of one second (end date of June 29 at 23:59:59 versus an end date of July 1 at 00:00:00) results in two monthly durations that differ by 1 day.

About building downward trends based on calendar periods

When you build a downward Trend based on calendar periods, to ensure accuracy you must couple it with a Forward Looking Inactivity component that uses end of calendar period. The reason for this requirement is explained by the following example.

Suppose you are looking for a drop of 50% in the total amount of deposits in a bank account in a calendar month for three successive months. Consider the case where an account had a drop of greater than 50% from a total of $3000 two months ago to a total of $1000 in the previous month. With two months already fulfilling the condition, the Trend is set up to evaluate to true if the account has a drop of greater than 50% in the current month (less than $500).

At the beginning of the current month, the account holder makes a single $100 deposit. If the Trend is evaluated now, it will see the total for the current month so far as $100 which is indeed more than 50% less than the previous month's $1000 figure and it will fire. However, because it is still early in the month, the account has not had the entire calendar month to fill up. Therefore, the Trend does not yet have all the three months of data and firing at this point would be incorrect.

To evaluate accurately, the Trend must wait until every deposit of the month has been made. Because evaluating every account at the end of the month is inefficient, it presents an interesting challenge to build the logic so that the detection is done correctly and efficiently.

This is where the end of calendar period FLI comes into play. Instead of connecting the Container Manipulator using this Trend to an Action, connect it to an FLI set to go off at the end of the month. The FLI introduces a delay in the evaluation that allows the logic to express: “This account is a candidate for a downward trend, but wait until the end of the calendar period in order to determine if indeed the downward trend is true.”

To accomplish evaluation, connect the FLI to a second Container Manipulator which uses the same Trend. When the FLI fires and the Trend attached to this second Container Manipulator evaluates true, then the downward trend is correct because it is the end of the month and no more deposits will be recorded for this current month. Using the FLI is also an efficient way to approach the detection because it restricts the end-of-month evaluation to only those accounts that are candidates for the downward trend.

So in general, if you are looking for a downward Trend for a calendar period, first place the Trend in a Container Manipulator and connect that Container Manipulator to an FLI set to go off at the end of that calendar period. Then connect the FLI to a second Container Manipulator, using the same Trend. The qualify-wait-test design pattern ensures that your logic efficiently detects a definitive drop within a calendar period. Note that the same pattern should be used for drops (downward spikes and ESDs) within a calendar period.

Note the following important detail when you build downward trends based on calendar periods. For an FLI configured to fire at the end of a calendar period, be sure to leave the **Set to the very next calendar period if the FLI is already set** checkbox clear, so that subsequent firings of the qualifying Trend (indicating that the partial deposit sum to date continues to meet the Trend condition) will set the FLI to the proper end-of-calendar period. If this option is selected, then the second arming of the FLI within the same calendar period would advance the FLI firing date to the end of the next calendar period. In this case, that is an unwanted behavior.

Because the FLI can be configured to only 1 end of calendar period, this method cannot be used when the calendar period length is greater than 1. For example, this method cannot be used if the trend periods are two calendar months long.

Why downard trends based on rolling periods are different

The precaution of using the end of calendar period FLI is not required if the periods in the Trend are rolling periods, because in that case the most recent period is already a complete period and the trend therefore does not need to wait additional time until all of its data is collected. It is only when the periods are calendar periods that the most recent period can be still incomplete and require an end of calendar period FLI to properly time the final trend evaluation.

About the trend type of the Trend component

The trend type of the Trend component detects either consistently high upward or downward changes in data values over a length of time. The length of time is fully divided into equal periods for which some calculated value in each period is either consistently increasing (or consistently decreasing) by a certain percentage, period to period across all periods.

To detect a trend, you must specify a field in the data source on which to do the trending. You must also specify the time period over which the trending is done and the percentage increase or decrease you are looking for. The data values all come from a designated field in a pre-populated container or select function.

The trend groups data values by fixed time intervals within the specified period. (For example, if a period spans three calender months beginning January 1 to the end of March, the data values are grouped into intervals for the months of January, February, and March.)

If the periods do not have subgroups (each period is not further divided into subgroups), then they are evaluated with a mathematical function (such as sum or average for example), and the resulting values are compared and the trend returns true if the percentage change of the values is consistently above the threshold value.

If you choose to use subgroupings within the trend period, Detect divides the trend period into subgroup time blocks, applies the subgroup function to the data in each subgroup, returns a value for each, and only then applies the selected trend period function on that set of returned values.

Note: In order for the component to detect an upward trend, set the trend percentage value to a positive value. To detect a downward trend, set it to a negative value.

Related tasks:

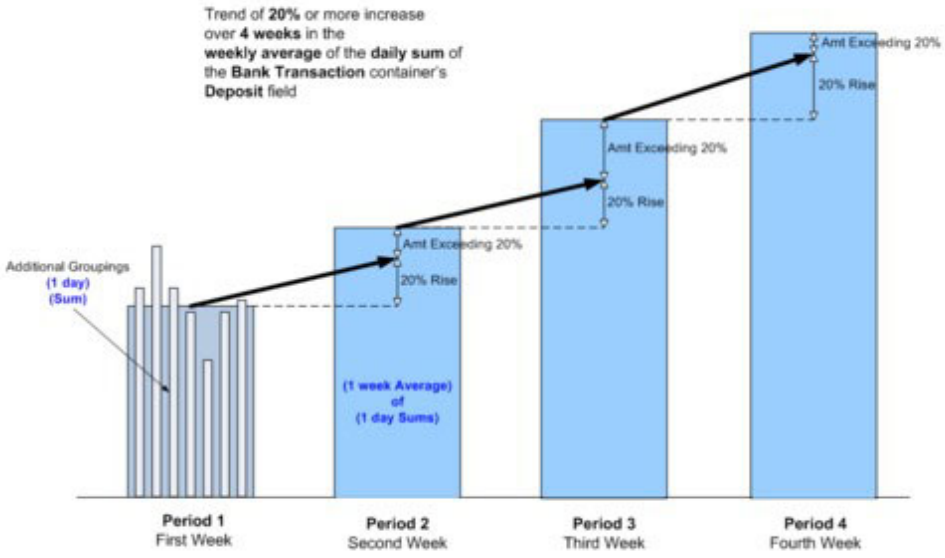
"To specify a Trend component" on page 81

Related reference:

"Advanced configuration options for the trend" on page 82

Example of a trend

The following figure illustrates data that would cause a trend to evaluate to true. In this example, the trend is looking for a 20% increase over four weeks in the weekly average of daily sum of deposits.



For this trend you would need a Container component with bank transactions (deposits, with date and amount). Then to build the trend you would do the following.

- Select a trend type (Trend).
- Set a percentage value (Constant, 20%).
- Select the Container component (Bank Transactions).
- Select a field upon which to watch for the trend (deposit amount).
- Select four trend periods.
- Set the trend period (1 week or 1 rolling week).
- Set the function type (Average).
- Select the subgroups option, then:
  - Select the subgroup time period (1 day).
  - Set the function (Average).
- Optionally, set Advanced Configuration options.

The options for the trend

The options for the trend allow you to define the following.

- The trend type, including the source of the trend percentage value.
- The data source upon which to observe the trend.
- The number of trend periods.

- The trend period value, which is a time period. By selecting the **Subgrouping with trend period** checkbox, you can also optionally establish how the time period is divided into blocks in which data values are grouped and evaluated by a function (such as sum or average).
- Advanced configuration options such as the first date on which the component can fire. For more information about Advanced Configuration options for the trend, refer to "Advanced configuration options for the trend" on page 82.

Adding a Trend component

Add a new Trend component as described in "Adding and editing components" on page 23. Specify the component properties as described in this section.

Note: Before you can create a trend, there must already be a container or select function defined so that the trend can use it as a data source.

To specify a Trend component

1. Define the **Percentage** value that represents the nature of the trend (percentage change) you are looking for.

   The value can be any of the following.

   - Positive, to track an increase
   - Negative, to track a decrease
   - A constant value
   - A lookup value from a data source
   - A calculated math value
2. In the **Data Source** area, select the data source by type and name, and identify the field on which to do the trending.
3. Enter the **Number of Trend Periods** you want to track. A trend must have at least two periods. (In the next step, you will define what each time period is.)

   The trend will only evaluate to true if each successive period increases (or decreases) over the previous period by the specified amount. For a tend with four time periods, the percentage of change must hold true for each of the three time period transitions: from period 1 to 2, 2 to 3, and 3 to 4.
4. Define the **Time Period Value** for the trend.

   For example, you might set the Time Period to be 3 calendar months long, and the Function to be maximum. In that case, the trend period value would be the maximum value in each three calendar month time period within the trend.
5. You can optionally set **Subgrouping within trend periods**.

   If you want to make the data function you are using act on subgroupings of values rather than on individual values in the container or select function, then select the checkbox **Subgrouping within trend periods** and set the **Subgroup Period** and **Function**.
6. You can optionally set **Advanced Configuration** settings for the trend. Advanced configuration settings can help to ensure that data has been tracked long enough, or that there are enough data points, to be statistically significant.
7. Click **Set Parameters** to save the details settings.
8. On the **Properties** tab, set the name and description of the trend.
9. Click **Set Parameters** to set the properties settings, click **Save** to save the component, then click **Close** to close the editor window.

Related concepts:

"About the trend type of the Trend component" on page 79

Related reference:

"Advanced configuration options for the trend"

Advanced configuration options for the trend

This section describes the optional advanced configuration settings for the basic trend. Advanced configuration settings can help to ensure that data has been tracked long enough, or that there are enough data points, to show be statistically significant.

Name of option	Description
Value of last uptrend period or first downtrend period >=	This option establishes the minimum value for the last upward trend period or the first downward trend period.
Number of container data points in each trend period >=	It is the minimum number of data points required within the each trend period for the trigger to fire. This setting ensures that there are enough tracked activities (such as deposits or withdrawals) to be statistically significant.
Number of non-empty subgroups in each trend period >=	This option establishes, for each trend period, the minimum number of subgroups that contain data. **Note:** This option only appears if **Subgrouping within trend period** was selected in the **Trend Period Value** section.
Calculate using zero values for empty subgroups	If this option is selected, zero values will be used in calculations for any empty subgroups. Otherwise, those empty subgroups will not be included in the larger calculation for the entire period. **Note:** This option appears only if **Subgrouping within trend period** was selected in the **Trend Period Value** section.
Entity start date must be before trend period begins	This option ensures that the system has been watching the entity for transactions for the entire historical time period. For each entity, their entity start date is the date of the first transaction captured by Detect for that entity.
Component will not fire until this date	This option sets the earliest date on which the component can evaluate to true. This setting allows you to ensure that the trend will not evaluate or fire until the data source (container or select function) that the trend is using has existed for a sufficiently long time and has sufficient data. Set this value based upon your data. For example, if you create a new trend that bases a calculation on 3 months of data then: • If you have ramped up the container with 3 months of data, you should set this date to the current date. That would allow the trend to start to evaluate to true immediately. • If you have just created the container today, and have not ramped it up, then set this date to the current date + 3 months. That setting forces the trend to wait for 3 months while the container builds up enough data.

Related concepts:

"About the trend type of the Trend component" on page 79

Related tasks:

"To specify a Trend component" on page 81

About the spike type of the Trend component

The spike type of the Trend component looks for a significant jump or a significant drop in comparison to a historical value.

Based on the parameters you set, the system formulates two values (Spike Value and Historical Value For Comparison) then compares the two values. The spike component returns true if the Spike Value is greater than or less than the Historic Value For Comparison by the specified percentage.

Note: To detect an upward spike, set the spike percentage value to a positive value. To detect a downward spike, set it to a negative value.

Related tasks:

"To specify a spike component" on page 84

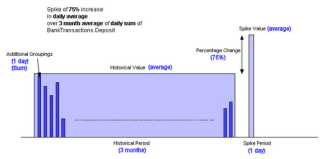
Related reference:

"Advanced configuration options for the spike trend" on page 85

Example of a spike component

The following figure illustrates data that would cause a spike component to evaluate to true. In this example, the spike component is looking for a 75% increase (spike) in the current daily deposit sum above the three month average of daily sums for deposits.

Note: Spike components are used as filters for Action components, Container Manipulators, and Qualifier components. They refine the triggering of those components to just those customers characterized by the parameters defined by the component.



To build a spike like this you would need a container with bank transactions (deposits, with date and amount). Then you would: do the following.

- Select a trend type (Spike).
- Set a percentage value (Constant, 75%).
- Select the container (Bank Transactions).
- Select a field upon which to watch for the spike (deposit amount).
- Set the spike value (sum for 1 day, either rolling or calendar).
- Set the historical value for comparison (average for three months). If rolling day were set for the spike period, then the historical period type would be automatically set to rolling months. If calendar day were set for the spike period, then the historical period type would be automatically set to rolling bounded months.
- Select the subgroups option, then:

  - Select the subgroup time period (1 day). If rolling day were set for the spike period, then the subgroup period type would be automatically set to rolling day. If calendar day were set for the spike period, then the subgroup period type would be automatically set to calendar day.
  - Set the function (sum).
- Optionally, set Advanced Configuration options.

The options for the spike component

The options for the spike allow you to define the following.

- The trend type (Spike), and the source of the trend percentage value
- The data source in which to look for the trend
- The spike value
- The historical value for comparison, which is a time period and a function applied to the values within that period. You can also optionally divide a historical period into subgrouping time blocks, define a subgroup function to apply to the data points in each subgroup, and only then apply the selected historical period function on that returned set of values.
- Advanced configuration options such as the first date on which the component can evaluate to true. For more information about Advanced Configuration options for the spike component, refer to "Advanced configuration options for the spike trend" on page 85.

Adding a spike component

Add a new Trend component as described in "Adding and editing components" on page 23. Specify the spike component properties as described in the following procedure.

To specify a spike component

Note: Before you can create a spike component, a container or select function must exist, so that the trend can use it as a data source.

These steps describe how to build a spike component.

1. In the **Type** drop-down on the Details tab list select **Spike**.
2. Define the **Percentage** value that represents the nature of the spike (percentage change) for which you are looking.

   The value can be any of the following.

   - Positive, to track an increase
   - Negative, to track a decrease
   - A constant value
   - A lookup value from a data source
   - A calculated math value
3. In the **Data Source** area, select the container or select function and identify the field on which to do the trending.
4. In the **Spike Value** area, define the spike period and function.

   For example, you might define the spike value to be the average for two rolling days.

5. Define the **Historical Value for Comparison**, which is the value to which the spike value will be compared.

   For example, you might define it to be the average over three weeks.
6. You can optionally set **Subgrouping within trend periods**.

   If you want to make the data function you are using act on the returned value from a second function applied to subgroupings of values rather than on individual values in the container, then select the checkbox **Subgrouping within trend periods** and set the **Subgroup Period** and **Function**.
7. You can optionally set **Advanced Configuration** settings for the spike trend. Advanced configuration settings can help to ensure that data has been tracked long enough, or that there are enough data points, to be statistically significant.
8. Click **Set Parameters** to save the details settings.
9. On the **Properties** tab, set the name and description of the spike trend.
10. Click **Set Parameters** to set the properties settings, click **Save** to save the component, then click **Close** to close the editor window.

Related concepts:

"About the spike type of the Trend component" on page 83

Related reference:

"Advanced configuration options for the spike trend"

Advanced configuration options for the spike trend

This section describes the optional Advanced Configuration settings for the spike trend. Advanced configuration settings can help to ensure that data has been tracked long enough, or that there are enough data points, to show be statistically significant.

Name of option	Description
Historical value >=	This option establishes the minimum value for the historical value against which the spike will be compared, Enter a value for the minimum historic average. The trigger will not fire for a customer whose historic base does not meet this value.
Rise or fall from historical value >=	This setting enables the trigger to track a fixed rise or fall over the historical base
Number of container data points in the historical period >=	This option establishes the minimum number of data points required within the each trend period for the trigger to fire. This setting ensures that there are enough tracked activities (such as deposits or withdrawals) to be statistically significant.
Number of non-empty subgroups in historical period >=	This option establishes, for the historical period, the minimum number of subgroups that contain data. **Note:** This option appears if **Subgrouping within trend period** was selected in the Historical Value for Comparison section.
Calculate using zero values for empty subgroups	If this option is selected, zero values will be used in calculations for any empty subgroups. Otherwise, those empty subgroups will not be included in the larger calculation for the entire period. **Note:** This option only appears if **Subgrouping within trend period** was selected in the Trend Period Value section.

Name of option	Description
Entity start date must be before historical period begins	This option ensures that the system has been watching the entity for transactions for the entire historical time period. For each entity, the entity start date is the date of the first transaction captured by Detect for that entity.
Component will not fire until this date	This option sets the earliest date on which the component can fire. This setting allows you to ensure that the trend will not evaluate or fire until the data source (container or select function) that the trend is using has existed for a sufficiently long time and has sufficient data. Set this value based upon your data. For example, if you create a new trend that bases a calculation on 3 months of data then: • If you have ramped up the container with 3 months of data, you should set this date to the current date. That would allow the trend to start evaluate to true immediately. • If you have just created the container today, and have not ramped it up, then set this date to the current date + 3 months. That setting forces the trend to wait for 3 months while the container builds up enough data.

Related concepts:

"About the spike type of the Trend component" on page 83

Related tasks:

"To specify a spike component" on page 84

About the exceeded standard deviations type of the Trend component

The exceeded standard deviations (ESD) type of the Trend component compares the value in a current time period to the average of that value over a historical time period.

The comparison is made in terms of a specified number of standard deviations, with the standard deviation itself being based on the historical assemblage of values. An ESD trend detects activity that exceeds the specified number of (positive or negative) standard deviations over specific time period.

This type of Trend component allows the detection of a jump or drop that is significant not just in its size but also in comparison to how much the value has typically jumped or dropped for this entity in the past.

Note: To detect an upward ESD spike, set the number of standard deviations value to a positive value. To detect a downward trend, set it to a negative value.

Related tasks:

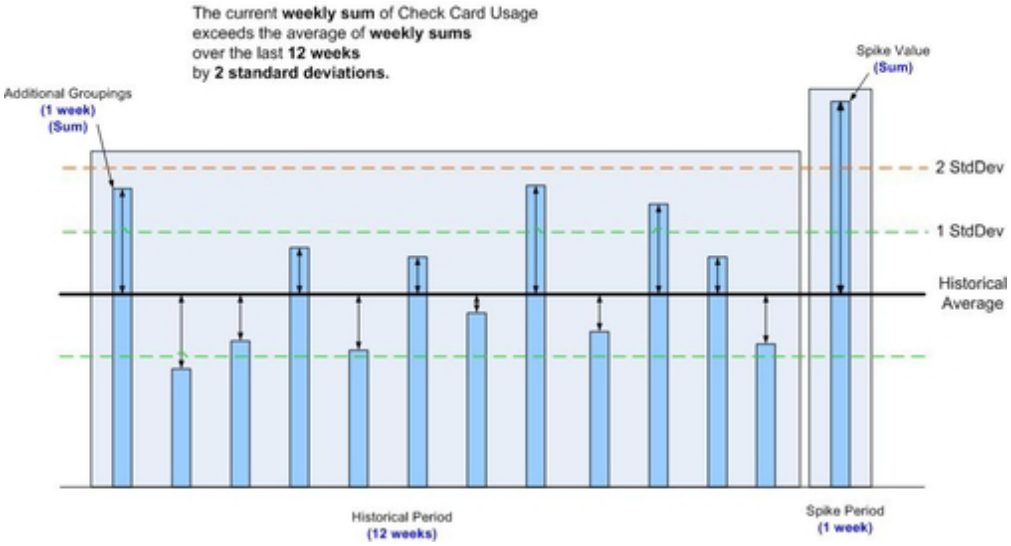
"To specify an exceeded standard deviations component" on page 89

Related reference:

"Advanced configuration options for the exceeded standard deviations component" on page 90

Example of an exceeded standard deviations component

The following figure illustrates data that would cause an Exceeded Standard Deviations (ESD) trend to evaluate to true. In this example, the ESD component was looking for the weekly sum of check card usage to exceed the average of weekly sums over the last twelve weeks by two standard deviations.



For this ESD you would need a container with check card transactions. Then to build the ESD you would do the following.

- Select a trend type (Exceeded Standard Deviations).
- Set a number of standard deviations (Constant, 2).
- Select the container (Check Card Usage).
- Select a field upon which to watch for the ESD (Amount).
- Set the spike value (sum for 1 week, either rolling or calendar).
- Set the historical value for comparison (12 weeks, the system automatically sets the function to average). If rolling week were set for the spike period, then the historical period type would be automatically set to rolling weeks. If calendar week were set for the spike period, then the historical period type would be automatically set to calendar weeks.
- Select the subgroups option, then set the subgroup valu.:
  - Set the subgroup time period (1 week). If rolling week were set for the spike period, then the subgroup period type would be automatically set to rolling week. If calendar week were set for the spike period, then the subgroup period type would be automatically set to calendar week.
  - Set the function (sum).
- Optionally, set Advanced Configuration options.

Difference between setting a spike period to a rolling week versus a calendar week

There is a difference in behavior when a spike period is set to a rolling week versus when it is set to a calendar week. When using rolling time, the periods will be completely full and span the entire week.

When deciding whether to use a calender week or rolling week for the spike value, keep these guidelines in mind.

- Use a rolling time period when you want to compare a complete period ending at the point of the transaction.
- Use calendar periods when you want to compare a period that is in the process of completion, to a fixed historical period.

A calendar comparison is more appropriate for uncovering abrupt changes; whereas a rolling comparison is more appropriate for changes that are smoothed into even divisions.

In the example above, the spike period was set to one rolling week. In that case, the week of the spike period is a full seven days long, ending at the current time. The spike value returned from that seven day period will be compared to the average of the twelve other rolling weeks which begin and end at the same time and day of the current transaction.

If a calendar week were used in the previous example, there would only be data in the current calendar week between the beginning of that week and the current transaction time (which would generally be a fraction of a week). The spike value returned from that calendar week period would be compared to the average of the twelve other calendar weeks which begin and end on strict calendar week boundaries.

The options for the exceeded standard deviations component

The options for the exceeded standard deviations (ESD) component allow you to define:

- The trend type (Exceeded Standard Deviations), including the source of the number of standard deviations
- The data source upon which to observe the trend
- The spike value
- The historical value for comparison, which is a time period. You can also optionally establish how the historical period is divided into blocks in which data values are grouped and evaluated by a function (such as sum or average).
- Advanced configuration options such as the first date on which the component can evaluate to true.

For more information about Advanced Configuration options for the ESD component, refer to "Advanced configuration options for the exceeded standard deviations component" on page 90.

Adding a new ESD component

Add a new Trend component as described in Addingand editing components. Specify the ESD component properties as described in the following procedure.

To specify an exceeded standard deviations component

Note: Before you can create a trend, there must already be a container or select function defined so that the trend can use it as a data source.

These steps describe how to build an exceeded standard deviations (ESD) component.

1. In the Type drop-down on the Details tab list select **Exceeded Standard Deviations**.
2. Set the **Number of Standard Deviations**.

   The **Number of Standard Deviations** value sets the number of standard deviations above (or below) the historical value for whatever time period you are looking at. The value can be a constant value or it can be custom to the customer based on a lookup value from a datasource or on a calculated math value.

   The value can be any of the following.

   - Positive, to track an increase
   - Negative, to track a decrease
   - A constant value
   - A lookup value from a data source
   - A calculated math value
3. In the **Data Source area**, select the data source by type and name, and identify the field on which to do the trending.
4. In the **Spike Value** area, define the spike period and function.

   For example, you might define the spike value to be the average of two rolling days.
5. Define the **Historical Value for Comparison**, which is the value to which the spike value will be compared.

   For example, you might define it to be the average over three weeks.
6. You can optionally set **Subgrouping within trend periods**.

   If you want to make the data function you are using act on subgroupings of values rather than on individual values in the container, then select the checkbox **Subgrouping within trend periods** and set the **Subgroup Period** and **Function.**
7. You can optionally set **Advanced Configuration** settings for the ESD trend. Advanced configuration settings can help to ensure that data has been tracked long enough. They can also ensure that you have enough data points, because the ESD does require a reasonable number of data points in the historical period for the concept of a standard deviation to be significant.
8. Click **Set Parameters** to save the details settings.
9. On the **Properties tab**, set the name and description of the ESD trend.
10. Click **Set Parameters** to set the properties settings, click **Save** to save the component, then click **Close** to close the editor window.

Related concepts:

"About the exceeded standard deviations type of the Trend component" on page 86

Related reference:

"Advanced configuration options for the exceeded standard deviations component"

Advanced configuration options for the exceeded standard deviations component

This section describes the optional Advanced Configuration settings for the exceeded standard deviations (ESD) component. Advanced configuration settings can help to ensure that data has been tracked long enough, or that there are enough data points, to show be statistically significant.

Name of option	Description
Historical value >=	This option establishes the minimum value for the historical value against which the spike will be compared. Enter a value for the minimum historic average. The trigger will not fire for a customer whose historic base does not meet this value.
Rise or fall from historical value >=	This setting enables the trigger to track a fixed rise or fall over the historical base.
Percentage rise or fall from historical value >=	This setting establishes a value that is the minimum percentage difference from the historical base. The trigger will only fire if the rise or fall exceeds this percentage and it meets the ESD condition as well.
Number of container data points in historical period >=	This option establishes the minimum number of data points required within the historical period for the trigger to fire. This setting ensures that there are enough tracked activities (such as deposits or withdrawals) to be statistically significant.
Number of non-empty subgroups in historical period >=	This option establishes, for the historical period, the minimum number of subgroups that contain data. **Note:** This option only appears if **Subgrouping within trend period** was selected in the Historical Value for Comparison section.
Calculate using zero values for empty subgroups	If this option is selected, zero values will be used in calculations for any empty subgroups. Otherwise, those empty subgroups will not be included in the larger calculation for the entire period. **Note:** This option only appears if **Subgrouping within trend period** was selected in the Trend Period Value section.
Entity start date must be before historical period begins	This option ensures that the system has been watching the entity for transactions for the entire historical time period. For each entity, the entity start date is the date of the first transaction captured by Detect for that entity.

Name of option	Description
Component will not fire until this date	This option sets the earliest date on which the component can fire. This setting allows you to ensure that the trend will not evaluate or fire until the data source (container or select function) that the trend is using has existed for a sufficiently long time and has sufficient data. Set this value based upon your data. For example, if you create a new trend that bases a calculation on 3 months of data then: • If you have ramped up the container with 3 months of data, you should set this date to the current date. That would allow the trend to start evaluate to true immediately. • If you have just created the container today, and have not ramped it up, then set this date to the current date + 3 months. That setting forces the trend to wait for 3 months while the container builds up enough data.

Related concepts:

"About the exceeded standard deviations type of the Trend component" on page 86

Related tasks:

"To specify an exceeded standard deviations component" on page 89

Chapter 15. Action Components

The purpose of an Action component is to notify the outside world that the specified behavior has occurred. It does so by writing data to a database, in the outcome table. For example, an Action component's outcome may be a recommendation to contact a customer due to a specified behavior.

An Action component writes an outcome to the database when it recognizes the occurrence of a single triggering event.

Events that can trigger an Action component

Any of the following event-generating components can trigger an Action component.

- Simple Event
- Pattern
- Forward Looking Inactivity
- Backward Looking Inactivity
- Container Manipulator

About the Action outcome

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

- 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
- An XML string containing a user-specified string message and as many relevant fields from the customer profile, transactions, existing containers, Select functions, and Join functions as the user chooses to specify in the Action.

By default, the outcome table is listed as a destination. Other destinations may be added (using the Configuration Utility) so that you can send the outcome from a particular trigger to a specified destination rather than to the default outcome table.

By default, the outcome table is listed as a destination. Other destinations may be added (using the Configuration Utility) so that you can send the outcome from a particular trigger to a specified destination rather than to the default outcome table.

The **Additional Information Details** option allows specific data from to be included with the outcome message. You can include data from these sources:

- Profile
- Transaction
- Container components
- Select Function components

Adding an Action component

Add a new Action component as described in "Adding and editing components" on page 23. Specify the component properties as described in this section.

The Action editor has four tabs.

- Component—The Component tab is where you specify the component whose triggering activates the Action component. See "To specify the component and outcome in an Action component."
- Outcome—The Outcome tab is where you specify the Outcome and Destination parameters. See "To specify the component and outcome in an Action component."
- Qualifier—The Qualifier tab is where you can optionally include a Qualifier component. See "To include a Qualifier component within another component" on page 38.
- Properties—The Properties tab is where you specify the name and other information about the component. See "About component properties" on page 24.

To specify the component and outcome in an Action component

1. In the **Components** tab:
   a. Select the component **Type**.
   b. Select the component by name.
   c. Click **Add Event** to add the selected component.
2. Select the **Outcome** tab to define the Outcome and Destination parameters.
   a. Select one or more outcome destinations from the list.
   b. In the **Outcome** box, enter an outcome text phrase to be sent to the selected destination(s).
   c. To add the outcome parameters to the Action component click **Add Outcome**. You can add more than one.
3. Optionally, define the **Additional Information** parameters on the Outcome tab.
   a. Click the checkbox for **Additional Information (optional)**.
   b. Select the **Type** of data to be included with the text message.

      The options you see depend on the type you selected.

      If you selected Profile Information or Transaction Information:

      - Select the actual **Datasource** from the drop-down list
      - Select a data source **Field** from the drop-down list

      If you selected **Container Information**:

      - Select a container from the **Container** drop-down list.
      - Select a field from the **Field** drop-down list.

        Selecting the asterisk (*) selects all the fields in a row of the container and filters the function choices to count and data.
      - Select a container function from the **Function** drop-down list.

        If you selected the asterisk for the field, you can select from a function. See "About functions" on page 27.

      If you chose **Select Function**:

      - Choose a select function from the **Select Function** drop-down list

- Select the **Field** and Function values in the same manner as those for a **Container**.

c. Click **Add to Outcome** to add the parameters to the outcome message. You can add multiple sets of parameters by repeating step 3.

4. Click **Save** to preserve your changes.
5. Optionally, include a Qualifier.
6. Click **OK** to save the Action component

Chapter 16. Publishing and Runing Trigger Systems

Before you can run trigger systems in your production environment, you should first publish them. There are two ways to publish a trigger system.

- You can publish the workspace that contains the trigger system.
- You can group workspace components together in a runset, and publish the runset. While only one workspace at a time can be run directly through the engine, you can reference components of multiple workspaces in a runset and run this set of components.

When you publish a runset or workspace, the triggers contained in the runset or workspace are available to be tested or run in your production system. Only one set of triggers can be published at a time.

About pub and xpb

Although you can run any workspace or runset from the command line, it is a best practice to run only a published runset or workspace as your production run.

When you publish a set of triggers from a workspace or a runset, Detect takes the set of triggers and does two things.

- It creates a read-only production workspace named `pub`, which overwrites any existing `pub` workspace. If you publish a runset, all of the triggers you selected for the runset are included in `pub`. There can only be one published workspace. You can view this workspace in the Workspace area of Detect, but you cannot edit it. This provides enhanced control of the data resulting from the run.
- It deactivates the existing pub workspace and archives it by saving it as xpb. This overwrites the existing xpb archive. There can only be one xpb workspace. If someone erroneously publishes a workspace or runset, you can restore the previous published set of triggers by publishing xpb. To preserve a copy of the components from the last xpb, you should copy or move those components into a new workspace before publishing a new workspace or runset.

Test runs versus production runs

To test the trigger systems in a workspace or runset, you run it using the Engine Manager area of the Detect web application.

To run a workspace or runset in your production environment, you must first publish it, and then run the Detect Engine from the command line.

For more information on testing and running pub, see the *IBM Unica Detect Administrator's Guide*.

Publishing trigger systems

You can publish a workspace from either the Workspaces manager or the Workspace Editor. You publish a runset from the Runsets area.

To publish a workspace from the Workspaces manager

1. Select **Workspaces** in the navigation bar.
2. Select the checkbox for the workspace you want to publish.
3. Click **Publish**.

   If you are publishing a workspace with a Backward Looking Inactivity component, you may see a dialog box that enables you to change the start date of the trigger.

   You can use the suggested start date or enter another one. Then click **Change Start Dates**.

To publish a workspace from the Workspace Editor

1. Select **Workspaces** in the navigation bar.
2. Select the workspace you want to publish.
3. Select the Properties tab.
4. Click **Publish**.

   If you are publishing a workspace that contains a Backward Looking Inactivity component, you may see a dialog box that enables you to change the start date of the trigger.

   You can use the suggested start date or enter another one. Then click **Change Start Dates**.

To publish a runset

1. Select **Runsets** in the navigation bar.
2. Select the runset you want to publish.
3. Click the Publish icon

Optional result of the task.

Optional postreq of the task - include if there is something the user must do next to complete the goal.

Chapter 17. Administering Components

The **Administer** feature on the Component List tab of the Workspace Editor allows users who have the Administer components XML permission to do the following:

- View, edit, and print component XML
- Override component locks
- Change component ownership

To access component administration

1. Click **Workspaces** in the navigation bar.
2. Click the name of the workspace that contains the component you want to unlock.
3. Click the **Component List** tab.
4. Select the component and click the Access Administration icon

About component locking

Components are automatically locked when they are being created or edited by a user. The locking mechanism prevents two users from accessing the same component at the same time. Once the component has been saved and no one is accessing it, the lock is removed and the component is accessible to other users. In some cases, the administrator may need to manually unlock a component to perform an emergency edit or other function. This feature should be used with care to avoid overwriting another user's work.

To unlock a component

1. Access the Administration window for a component as described in "To access component administration."
2. Click **Unlock**.
3. If the component is currently in use by another user, the **Lock Status** is LOCKED by username and the **Unlock** button is enabled.

   If the component is not locked, the Unlock button is disabled.
4. Click **Unlock** if it is appropriate to unlock the component.

   A warning message states that unlocking the component may cause the other user to lose his/her work.
5. Click **OK** to unlock.
6. Click **Close** to close the Administration window.

About component ownership

The user who writes a component becomes its owner by default. If, at a later time, a user with a user level higher than the original owner modifies the component, ownership of the component transfers to the higher level user who modified it. Users with lower levels do not have permission to view or modify components that are owned by users with higher levels.

If a higher level user modified and became the owner of the component, the lower level user would no longer be able to work with the component. To reassign ownership to the lower level user, an administrator must change the component ownership.

To change the owner of a component

1. Access the Administration window for a component as described in "To access component administration" on page 99.
2. If the component is locked, unlock it.
3. Select the new owner from the drop down list.
4. Click **Save Owner**.
5. Click **Close** to close the Administration window.

Working with a component's XML

Although the Detect interface provides the most frequently used method for generating components, there are cases when an administrator will want to handle the XML directly. However, it is always best to create and edit components using the Workspace Editor interface to ensure that the component will perform as expected within the Detect Engine.

Note: Only administrators very familiar with Detect and the way it uses XML to represent triggers should modify the XML directly.

To view and print the component source

This function provides a view of component XML in read-only mode.

1. Access the Administration window for a component as described in "To access component administration" on page 99.
2. Select the **Source** tab.
3. The XML is displayed in read-only mode.
4. Click **Print** to print a hard copy of the XML.

To edit the component source

1. Access the Administration window for a component as described in "To access component administration" on page 99.
2. Select the **Source** tab.

   The XML appears in read-only mode.
3. Select **Edit**.

   If the component is currently in use by another user, the **Edit** option is not available.
4. Edit the XML as needed.

   You can reset the XML to its original state by clicking Reset.

5. Click **Save** to save your changes.

   Clicking **Save** will automatically validate that the component XML is internally consistent.
6. Click **Close** to close the Administration window.

To verify that component XML is internally consistent

The XML Validate feature verifies that the XML you have entered meets the basic criteria for well-formed XML (that is, it conforms to the XML syntax rules). The XML Validate feature does not check the XML against the criteria defined in the Detect schema for that XML document or verify that the revised code will function as intended within the Detect Engine.

1. Access the Administration window for a component as described in "To access component administration" on page 99.
2. Select the **Source** tab.

   The XML is displayed in read-only mode.
3. Select **Edit**.

   If the component is currently in use by another user, the **Edit** option is not available.
4. Edit the XML as needed.
5. Click **Validate**.
   - If your edit was valid, a confirmation message states **Your Code is Valid**. Click **OK** to close the confirmation message.
   - If the edit was invalid, an error message appears with an option to view error details. Choose one of the following actions from the window.
     - Click **Error Details** to display additional information about the error.
     - Click **Continue** to close the error window.
     - Close the error window using the Windows controls.
6. If necessary, update the XML to remove the errors and repeat the validation process or click **Reset** to return the XML to its original state.
7. To exit the editor without making any change to the XML, click **Close**.
8. Click **Save** if you want to save your changes.
9. Click **Close** to close the Administration window.

Appendix. Sample Trigger Systems

The examples in this section represent complete trigger systems designed to recognize specific customer behavior. The components that make up a trigger system are organized in a hierarchy. The highest level component in the hierarchy is the Action component.

Goal: Identify customers who make a printer purchase

ACTION COMPONENT

- Name is "Customer Purchased Printer Action"
- Write the message "customer purchased printer" and the Store ID to the outcome table when the Simple Event "A Printer Purchase" is triggered.

SIMPLE EVENT COMPONENT

- Name is "A Printer Purchase"
- Trigger this Simple Event when the following is true of a sales detail transaction:
  - transaction type = purchase AND
  - sku = a printer
- When triggered, record Store ID

OUTCOME

- Message = "customer purchased printer"
- XML includes the Store ID

Goal: Identify customers who make 3 large purchases in a week

ACTION COMPONENT

- Name is "Good Customer Made 3 Large Purchases in Week"
- Write the message "3 LG purchases in week for Good customer" to the outcome table when:
  - The Pattern component event "3 Large Purchases in 1Week" is triggered AND
  - The Qualifier "Good Customer" is triggered "is true"

QUALIFIER COMPONENT

- Name is "Good Customer"
- When the quality ranking on a customer's profile = "good," trigger this Qualifier "is true"

PATTERN COMPONENT

- Name is "3 Large Purchases in 1Week"
- Trigger this Pattern component event when the Simple Event "A Large Purchase" is triggered 3 times in 1 week

SIMPLE EVENT COMPONENT

- Name is "A Large Purchase"
- Trigger this Simple Event when the following is true of a sales detail transaction:

– transaction type = purchase

AND

– the total dollar amount > $335

OUTCOME

- Message = "3 LG purchases in a week for Good customer"

Goal: Identify customers who make 3 large purchases in a week, where the total of those purchases is more than $1000

ACTION COMPONENT

- Name is "Good Customer Made 3 Large Purchases in Week Totaling more than $1000"
- Write the message "3 Large Purchases Total more than $1000" to the outcome table when:
  – the Pattern component event "3 Large Purchases Total more than $1000" is triggered

  AND

  – the Qualifier "Good Customer" is triggered "is true"

QUALIFIER COMPONENT

- Name is "Good Customer"
- On a customer's profile, when the quality ranking = "good" trigger this Qualifier "is true"

CONTAINER MANIPULATOR COMPONENT

- Name is "3 Large Purchases Total more than $1000"
- Trigger this Container Manipulator event and clear the contents of the "Purchase Amounts" container when:
  – The Pattern component event "3 Large Purchases in 1 Week" is triggered
  – The sum of the amounts in the "Purchase Amounts" container > $1000

PATTERN COMPONENT

- Name is "3 Large Purchases in 1Week"
- Trigger this Pattern component event when the Simple Event "A Large Purchase" is triggered 3 times in 1 week

SIMPLE EVENT COMPONENT

- Name is "A Large Purchase"
- Trigger this Simple Event when the following is true of a sales detail transaction:
  – transaction type = purchase

  AND

  – the total dollar amount > $335 CONTAINER COMPONENT

CONTAINER COMPONENT

- Name is "Purchase Amounts"
- Holds data of the "money" type
- Allows room for 3 occurrences data
- Saves the data for 1 week

OUTCOME

- Message = “3 Large Purchases Total more than $1000”

Notices

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

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

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

IBM 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.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites 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.

Printed in USA