IBM Unica Detect
Version 8 Release 5
July 31, 2012

Administrator's Guide

Note

Before using this information and the product it supports, read the information in "Notices" on page 173.

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 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 to gather

Before you contact IBM Unica technical support, 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."

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 the installation directory for each application.

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. System Overview

This chapter provides a broad overview of the setup, configuration, and operation of the IBM Unica Detect application.

About IBM Unica Detect

Detect allows your business to trigger targeted actions in response to complex patterns of activity discovered in incoming streams of transactions. You can use the following Detect features to improve your business:

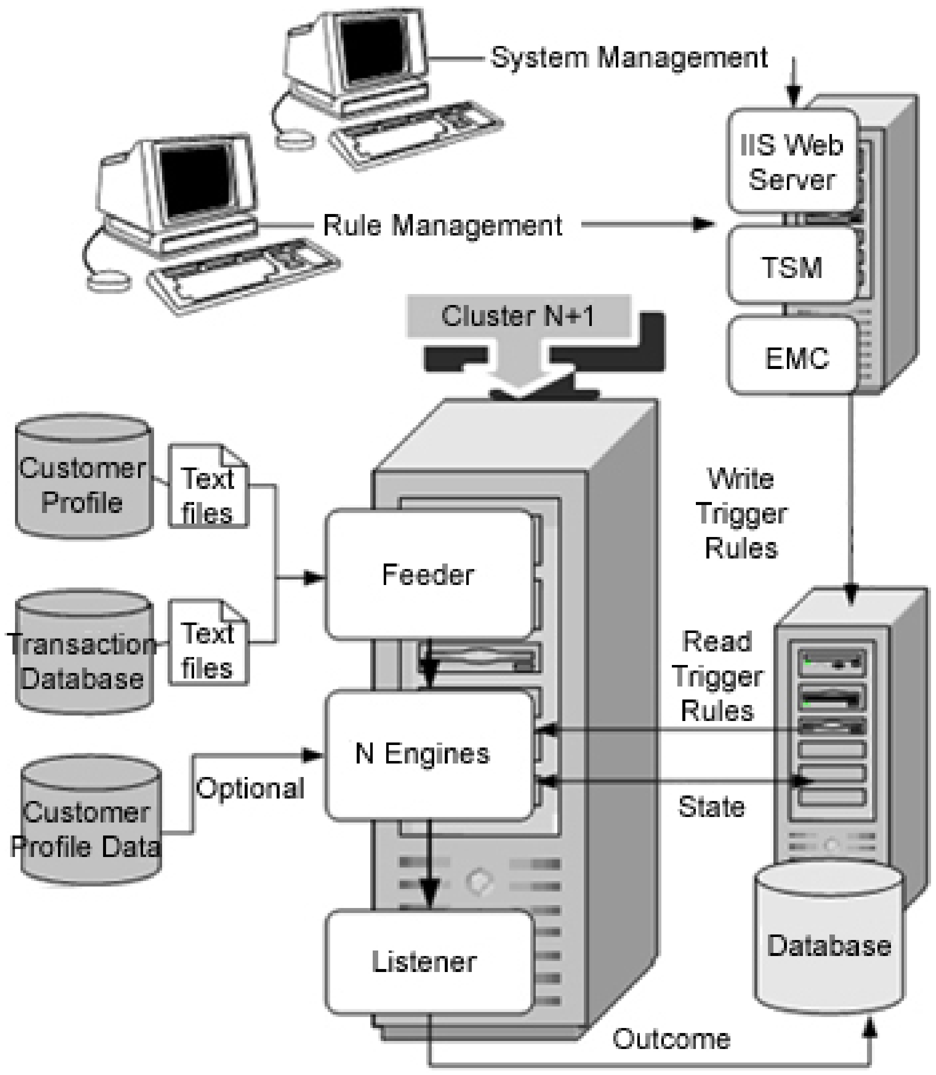
- Make decisions on incoming data quickly
- Configure the system to receive any kind of transactional input and to access existing static data
- Create and modify the central components that control the triggering behavior using an intuitive web-based interface

After you read this chapter you will have an overview of the following topics:

- Description of the Detect environment configuration
- What the administrator does to define data sources and users
- How the user defines trigger systems and their components
- How the Detect engine runs

About the Detect environment

The figure below shows an example of a Detect deployment environment.



The trigger **Engine** is the central component of Detect. The engine can be explicitly run but usually is run at specified intervals (for example, nightly) by a scheduler. When Detect starts, it reads a set of configuration parameters from the Detect database. These parameters specify where the system can find all of its components as well as the **trigger systems** (or rules) that define the patterns to match, and where to look for the incoming data.

The engine analyzes data from any or all of batch transactions and customer profiles. During a run, the engine uses the detection components to analyze the incoming data for specified patterns. When the engine finds matching patterns in the data, it fires an action and sends the specified outcome to a message queue so that the Listener can put them in the database. The contents of the Outcome database can be fed into other marketing and sales systems used by the organization or used for reporting.

About the Detect functional units

From the point of view of machine resources, Detect can be grouped into functional units. You can modify which components run on which unit but the relationships among the components makes certain configurations more efficient than others. In this list, the key components are grouped by the unit on which they most commonly reside:

- **Web Server Unit —** Enables web access to the user interface pages for system configuration and trigger building, testing, and deployment.
- **Cluster Unit —** Provides system processing. Each cluster unit has its own Feeder, Outcome Listener, and Engine components to eliminate possible input/output bottlenecks. Within each cluster unit, one or more engines process transactions using rules to search for matches to defined triggers and make heavy use of memory and CPU. The Feeder reads transactions from the data feed files and makes heavy use of input/output resources.You can set up Detect to run with multiple cluster units installed on multiple machines, as described in “About scaling with multiple cluster units.”
- **Database Unit —** Holds the Detect database. The database contains metadata for the system such as configuration information, rules, logging info, system state settings, and customer state. The database unit usually contains the outcomes that result from the triggers fired by the engine. The database unit makes heavy use of disk storage.

About configuration options

Detect is installed on each of the machines that run Detect components. You can specify the components that run on a particular machine using the Configuration Utility.

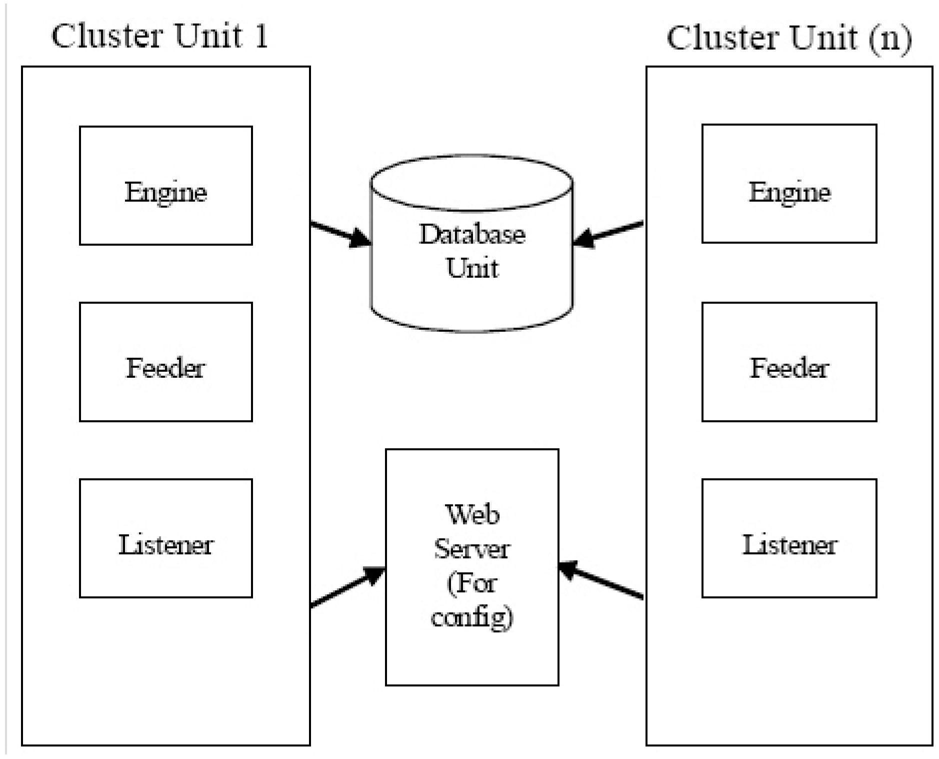
Configuration options are very flexible. You can set up all of these components on a single machine or use a configuration of multiple machines to optimize system performance. As the processing requirements of your application increase, so can the number of cluster units included in your configuration. Also, you may decide to combine two functional units onto one machine. The final decisions on how to set up your system should take into account your system resources as well as the throughput needed by your system.

About scaling with multiple cluster units

Most enterprise Detect implementations use a single cluster unit, but customers with large transaction volumes or tight processing windows may deploy multiple clusters units on multiple machines. In a Detect environment with multiple cluster units, the cluster units typically share a single database and a single web server unit consisting of a Trigger Set Manager (TSM), an Enterprise Management Controller (EMC), and a single IIS instance.

About multi-cluster configuration

The diagram below is an example of how a high volume customer might configure the different Detect components and units on hardware servers for increased performance. In this example, sets of Feeders, Engines, and Listeners are configured on a single hardware server known as a Cluster Unit. The Database and the Web Server would each be on their own hardware servers. To increase performance in this environment a customer would add an additional cluster units which house additional sets of Feeders, Engines, and Listeners.



If even greater performance were needed, more cluster units could be added.

About system configuration

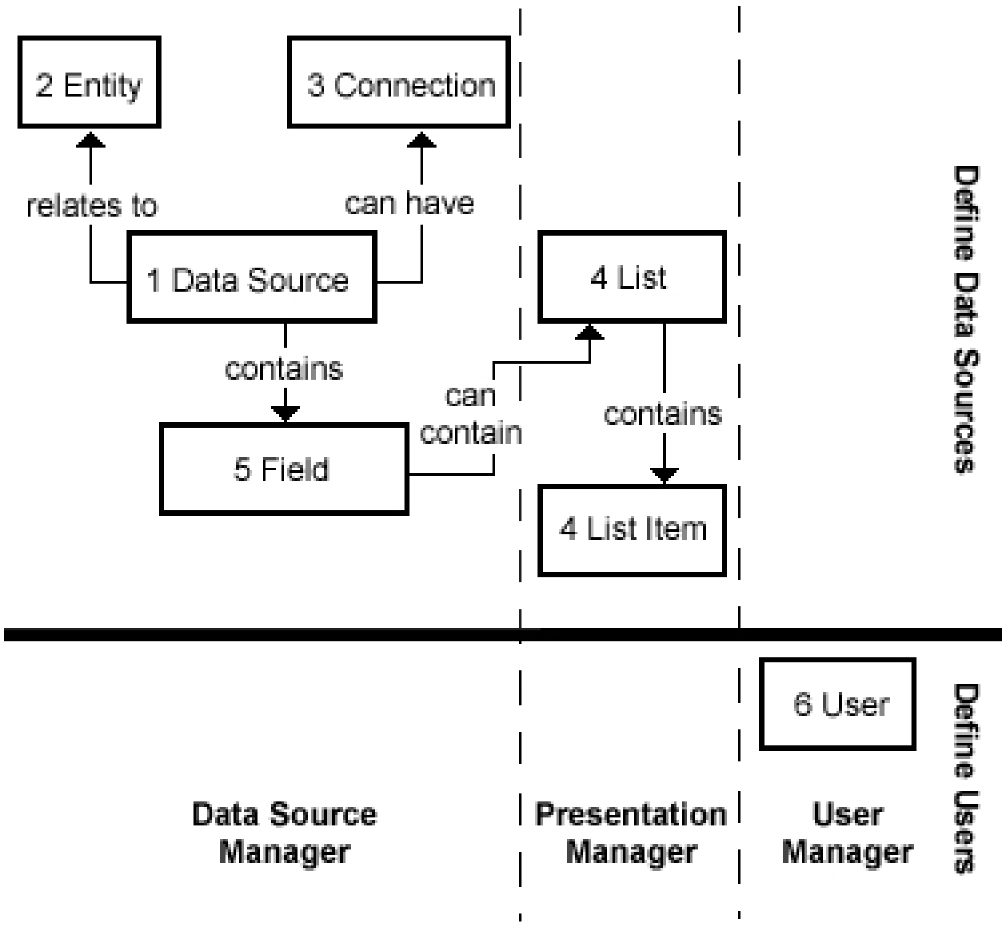
After installation, the system is configured as follows:

- The System Administrator defines the location and contents of the data sources.
- The System Administrator sets up business users who will define the components and trigger systems.
- Business users assemble components and trigger systems that are used to analyze the data from the data sources and decide outcomes based on the analysis. (This chapter provides overview information; for details about components and trigger systems, refer to the *Detect User Guide*.)

The system administrator and business users perform these tasks using a set of tools accessible from a web-based application. The information entered during configuration is saved in the Detect database in the rule schema. The G_RUN table shows the names of the database tables that save information about each object.

About administrative setup

The administrator defines the data sources the system will read and also sets up the users of the system. The following diagram shows a general outline of the objects created and the utilities the administrator uses to configure the system.



Administrative Setup (Overview)

The numbers in front of the objects in the previous figure show the order in which they would logically be created based on their dependencies. The following table describes the process that the administrator takes to configure the system and describes the objects shown in the previous figure that define the characteristics of the system.

Table 1. Data Sources Manager

1	Define the **Data Sources—** describes the sources of data. A data source has a specified entity type, and a connection if it is a database source. For example, for a financial institution there might be one data source for stock transactions and one for counter transactions.
2	Create the **Entity Types—** defines identifying characteristic of a data source. Examples of entity types are customers and accounts. The data in a data source is sorted according to the ID assigned to the entity type. Entity type is represented using a single alphabetic character such as *c* for customer. While you are defining a Data Source, you can use the Data Sources Manager to add an entity type. You can add entity types at any time using the Configuration Utility.

Table 1. Data Sources Manager (continued)

3	Define the **Connections—** specifies how the data source is connected to the Detect engine. Connections are needed only for data sources that come from database tables. While you are defining a Data Source, you can use the Data Sources Manager to add a connection for the data source. You can add connections at any time using the Configuration Utility.

Table 2. Presentation Manager

4	Each data source has fields associated with it. When you define a field, you specify the type of value that it can contain. Some fields can be assigned items from a specified list of values. You define the Lists associated with Fields and the List Items that they can contain in the Presentation Manager. For example, a data source containing bank transactions might have a field called *TransactionType*. This field is associated with the list called *TransactionTypeList*. The list contains the list items *deposit*, *transfer*, and *withdrawal*.

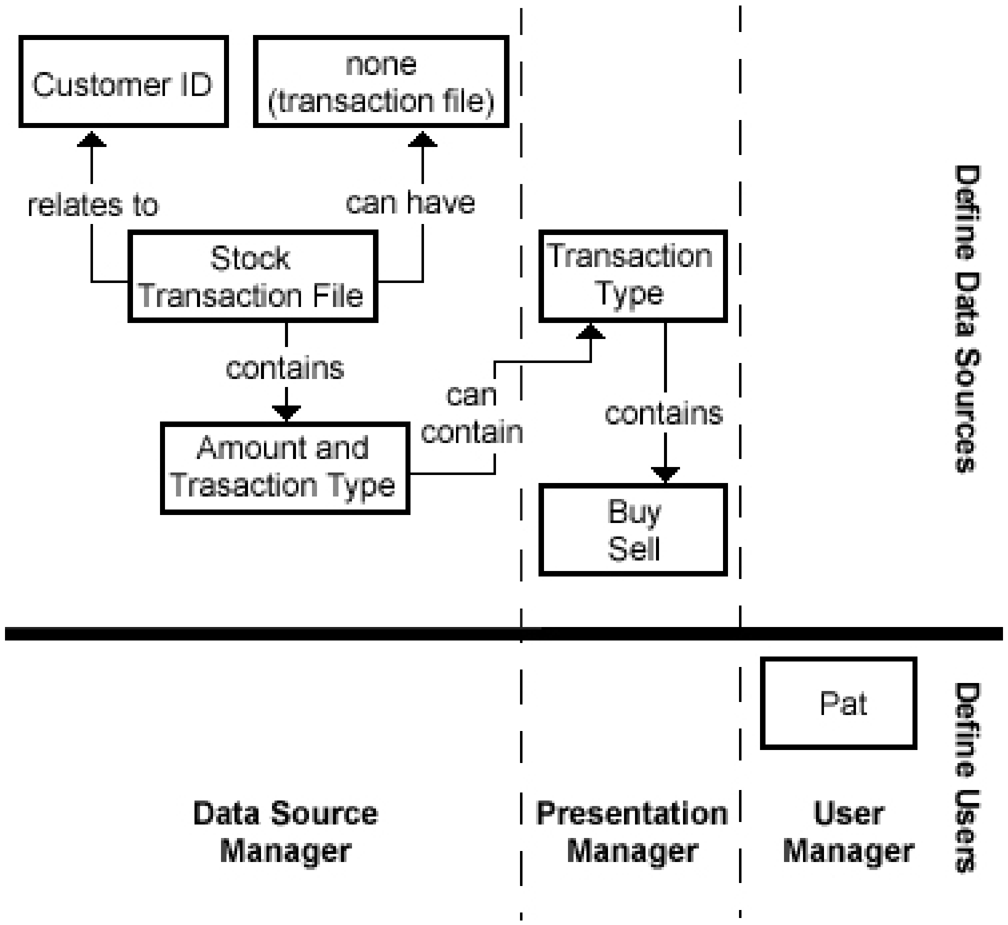
Table 3. Data Sources Manager

5	Define the **Fields** associated with each data source—Once all of the fields that use lists have been defined, the administrator can return to the Data Source Manager and describe all of the fields that belong to each data source. For example, the data source *CounterTransactions* might have the fields *CustomerID*, *Date*, *TransactionType*, and *Amount*.

Table 4. User Manager

6	Define the **Users—** business users of the system who develop the components and trigger systems that define the outcomes of the data.

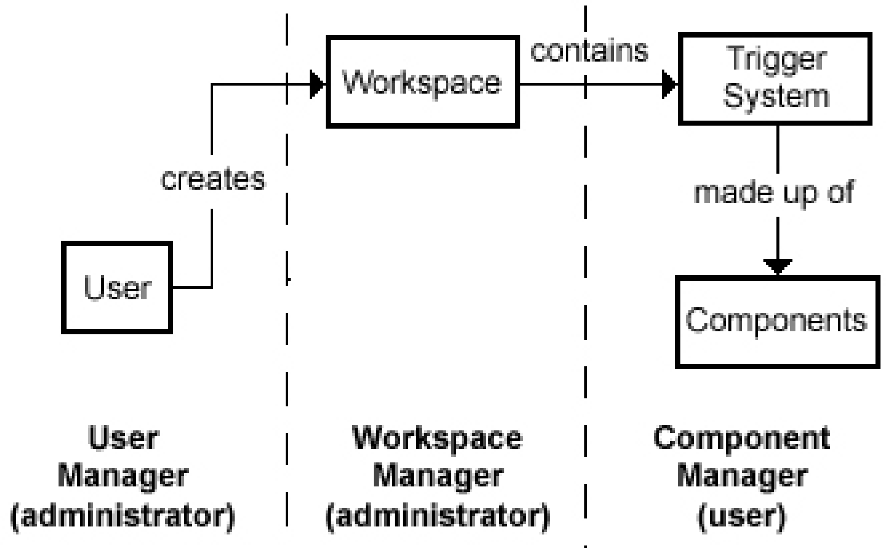
The following figure shows the abstract objects set to values, to provide an example of the relationships:



In this example, the user is **Pat**. The data source is the **Stock** transaction file. This file is a text file that contains the fields **Amount** and **TransactionType**. **TransactionType** is a List that contains the items **Buy** and **Sell**. The **Stock** transactions file uses the entity **CustomerID** as the primary key for the transactions. Because it is a transaction file and not a database file, there is not a connection associated with the file.

About designing the trigger systems

Once the administrator has setup the data sources and created users, the users are ready to define the trigger systems that the engine will use to analyze the incoming data. The following figure shows the objects involved in the creation of the trigger systems.



The administrator creates **workspaces**, which hold the trigger systems (rules) that business users enter into the system. Workspaces provide a way to test and experiment with trigger systems. When a workspace is considered ready to be put online, it is promoted to pub, the production workspace. An administrator can create as many workspaces as needed.

A workspace contains trigger systems. A **trigger system** represents a business rule. Here is an example of business logic that could be represented in a trigger system:

If the customer makes three purchases within one month and at least one purchase is for an item over $500, send the customer a $25 gift certificate.

The trigger system is made of **components**. A component uses other components as building blocks to create complex decision systems that read and operate on data from the transaction files and databases defined as data sources for the application. There are several different types of components, and they function in very different ways. For more information about creating components and developing trigger systems, refer to the *Detect User Guide*.

About testing a workspace and running the engine

After you create a trigger system with its underlying components, you should run the system to test it for both correctness and effectiveness. To run the trigger systems in a workspace, you do the following.

- Set up test transaction files that have the same formats as the ones that the system will use.
- Use the Engine Manager to set up a run of the workspace.

All of the transaction files that the Engine reads belong in the specified **Feed Directory**. Even though a workspace can contain components that relate to more than one entity, you can perform a run for only one entity at a time. When you run a workspace using the Engine Manager, you can monitor engine processing for the run and view statistics for the firing of the components.

After you test and tune the components in a workspace, you promote the workspace so it becomes the **published** workspace, named *pub*. The published workspace is the workspace that is regularly run as part of the running of the production system.

While runs for user-created workspaces can be started directly from the Engine Manager, runs for the *pub* workspace must be initiated using the command line. This setup prevents unintended runs on the production environment, which would add unwanted data to the state history that is generated as a result of a run. In a production environment, the engine is generally scheduled to run at a specified time.

Chapter 3. Quick Start

After you install IBM Unica Detect and configure the environment, you are ready to get started using Detect.This chapter provides a quick overview of each of the steps that are necessary to configure Detect.

Log in to IBM Unica Detect

1. Enter the Detect URL in the address bar of Internet Explorer. The URL is in the following form: `http://`*`machine_name`*`/`*`IIS_directory_name`*, where *`machine_name`* is the name of the server machine and *`IIS_directory_name`* is the name of the virtual directory created for Detect at installation time.
2. Disable popup blocking in the browser.

   Note: Pop-ups are used throughout the system. If you do not disable popup blocking on all browsers used to access the system, you may not be able to use all of the features.
3. Set the browser to allow cookies.
4. At the login screen, enter an administrator login name and password. Then click **Sign In**.

   An administrator account named **system** was created during the installation process; others can be created using the **system** account or another account with appropriate permissions. The default login for the **system** account is **host**, although this password may have been reset.
5. If you want to change the admin user's password now, do the following.
   a. Click **Users** then select the admin user and click **Edit**.
   b. Give the admin user a new password and click **Save**.

Define the data sources

These are the basic steps for creating a data sources.

1. Click **Data Sources** in the navigation bar.
2. Click **Add**.
3. Select the appropriate data source type from the **Source Type** drop-down list.
4. Select or enter information in the fields.
   - **Logical Name –** Enter an identifier in the field.
   - **Display Name –** Enter a descriptive name in the field.
   - **Description –** If desired, provide information about the data source in the field.
   - **Source Name –** Enter the name of the feed file.

   The Source name cannot contain spaces. For maximum clarity, it is recommended that it be set to the same string assigned above to Logical Name, although that is not required by the application.

   The format is `unica.`*`<entity type>`*`.`*`<source name>`*`.`*`<YYYYMMDDhhmmss>`*

   Note the following.
   - The prefix used in the source name can be any of these values: `unica`, `detect`, `insights`, `marketsoft`, `elity`. The example above uses `unica` as the prefix.
   - The entity type is always a single letter.

- The time portion of the datestamp is optional, but provides greater granularity.

For example, if the source name is profile, the feed might be named:

```
unica.a.profile.20061001
```

Depending on what Source Type you selected, you may also see one or more of the following fields.

- **Entity Type Name –** From the drop-down list, select the entity that you will track in the data source. If the entity type you need is not listed, click the **Add Entity Type** link and then add it. An ***entity type*** defines the characteristic of a data source. For example, you can define a customer entity type or perhaps an account entity type. When you define an entity type, you also provide that entity type with an ID. The ID is important, because the data in a data source is sorted according to the ID assigned to the entity type. Entity type is represented using a single alphabetic character, such as *c* for customer.
- **Use Default Profile –** Select this option if you want the system to include default profile values for transactions that come in without profiles.
- **Preload –** Select **None** if you did not preload any data into memory at runtime and **Full** if you did preload data into memory at runtime.
- **Connection –** Select the name of the connection. If the connection is not listed, click **Add Connection** to add it. Connections are required for data sources that are defined as either Database Tables or Lookup Tables.
  - **Connection Name –** the name used internally by Detect. The maximum length is 30 characters.
  - **Connection Description –** a descriptive name for the connection, one a user can understand.
  - **Provider –** select the provider of the database from the drop-down list.
  - **Server –** the name of the physical database server. This information is required only when the provider is set to SQLServer.
  - **Database Name –** the name that was given to the database service.
  - **User Name –** the user name that is used to log onto the database.
  - **Password** and **Re-enter Password –** the password associated with the supplied user name. The password must be re-entered to detect mistakes.

5. Click **Save**.
6. Repeat steps 2 through 5 for each data source that you want to create.

Define the lists associated with data source fields

Note: If you do not intend to define any Data Source Fields that can be assigned a list value, you can skip this set of steps.

Each data source has fields associated with it. When you define a field, you specify the type of value that it can contain. Some fields can be assigned items from a specified list of values. You define the **Lists** (or *domain tables*) associated with Fields and the **List Items** that they can contain in the Presentation Manager. For example, a data source containing bank transactions might have a field called TransactionType. This field is associated with the list called TransactionTypeList. The list contains the list items deposit, transfer, and withdrawal. The list consists of items with both a logical name, indicting the way that data element appears in the incoming data stream, and the display name which shows how it will display in a drop-down list used to select from the various allowed values of the field within the Detect user interface.

You can assign lists to data source fields. Once you define a list, you can add the individual items in the list.

Follow this procedure to create a lists that you want to assign to data source fields:

1. Click **Presentation** in the navigation bar.

   The screen shows you a tree structure of the existing lists on the left side of the page. The right side of the page will be empty until you choose an operation.
2. Click **Add List**.
3. Update the **List Name** and **List Description** fields.
4. In the **Creation Method** area, choose the appropriate radio button to either create an empty list (**Create Empty**) or create both the list and the list items based on the contents of an existing file (**Create with Content**).

   Note: When using **Create with Content**, please note that the importer reads `.csv` files. You should organize these .csv files so that the contents are expressed as "logical name", "display name". The "display name" text appears as the list items on the left side of the screen.
5. If you selected **Create Empty**, click **Add**. The left side of the window shows your new list, in alphabetical order.

   The **Browse** button is enabled when you select **Create with Content**. Once you do so, you can follow this procedure to use content from an imported file to create both the list and the list items in a single step:

   a. Click **Browse** to display the file chooser.

   b. Select the file from which you want to import content and click **Open**.

   c. Click **Add**.

   The left side of the window updates to show your new list along with the items in that list. (Lists and list items are presented in alphabetical order.)

To add items to a list

If you created an empty list, you need to add the list items.

1. On the left side of the list, click on the name of the list to which you want to add items.
2. Click the **Add List Item** button.

   The right side of the window is updated prompting you to enter a logical and display name for an individual list item.
3. Enter both the logical and the display name for the list item and click **Add**.

   The new item is added to your selected list and the right side of the page changes to edit mode allowing you to make changes to your newly added item.
4. Repeat these steps for each item you need to add.

Add fields to a data source

Now that you have created all the Lists that might be associated with Data Source Fields, you can return to the Data Source Manager and add the fields to each data source.

1. Select **Data Sources** to display the list of existing data sources.

2. Click the name of the data source to which you want to add fields or select the radio button next to the data source and then click **View Fields** to display the data source fields page.

   The screen updates to show the field editing controls and any existing fields. (The window will contain a message if no fields exist.)
3. Click **Add Field** to display the field editor window.
4. Enter the appropriate information in the **Logical Name**, **Display Name**, and **Description**.
   - **Logical Name** is the name used by Detect to recognize the data coming in on this field. The maximum length is 30 characters. Do not use any leading spaces.
   - **Display Name** is the user-friendly name for the data contained in this data field. This is the name displayed in the component editors. The maximum length is 30 characters. Do not use any leading spaces.
   - **Description** is an optional field to provide information about the data field. The description is seen only by the administrative user.
5. Select the **Data Type** from the drop-down list.

   The value defines the type of data the field contains, such as *String*, *Number*, *Date*, *Money*, or *Boolean*.
6. Select the **Display Type** from the drop-down list.

   This information is used within the Detect editors to control the user entry options for the field. You can specify either *Text*, *Date*, or *List*.

   Note: If the Display Type is **List**, then you must select a Domain Table that contains the values that are allowed for the field. Domain tables are lists defined in the Presentation Manager.
7. Select the **Field Application Type** from the drop-down list.
8. Click **Save**.

Create business users

Business users develop the components and trigger systems that define the outcomes of the data.

1. Click **Users** the navigation bar to display the list of existing users.
2. Click **Add**.
3. Enter a name in the **User Name** field.
4. Enter a password in both password fields (**Password** and **Retype Password**).
5. Enter an integer value in the **User Level** field to create a relative user level for this user.

   The user level that you enter here is evaluated against the values entered for other users to create a relative permission level. Users with a larger integer value have "greater" permissions as they could be set to have read, edit or delete permission on items created by users with a lower permission value.
6. Use the checkboxes in the **Permissions** area of the screen to specify whether this user has read, create, edit or delete permissions for their own components or for components owned by others with lower user levels.
7. Also set whether the user has **Other permissions**, such as being allowed to administer components XML.
8. Click **Save**.

Create the type descriptors

Here are the basic steps for defining type descriptors.

1. Select the **Type Descriptors** in the navigation bar to display a list of existing type descriptors.
2. Select **Add New** in the navigation tab area to display the type descriptor fields.
3. If the type descriptor is for a container, check the **For Container Use** option at the top of the screen.

   The screen will refresh to change the available options.
4. To add a field to the type descriptor, fill the **Field Attribute** settings and click **Add Field**. Add as many fields as needed.
5. To save the type descriptor click **Save Definition**.

   You are returned to the list of existing type descriptors.

Create the workspace

There are two ways you can create a new workspace. You can copy an existing workspace from the workspace list, or you can add a new, empty workspace and optionally copy components into it. This procedure describes the second method.

1. Select **Workspaces** in the navigation bar.
2. Click **Add**.
3. Enter a descriptive name for the workspace in the **Name** field.
4. Enter a unique identity code for the workspace in the **Code** field.

   The workspace code must be 3 upper-case alphabetic characters (using the English alphabet, even if you are using multi-byte characters elswhere in Detect).

   You must remember this code in order to refer to the workspace when configuring other Detect features.

   Note: Runspace Codes are case sensitive, so T01 is not the same as t01.
5. Optionally, you can copy components from an existing workspace into this workspace. Click **Copy Components From** then select a workspace from the drop-down list, then select the components you want to copy, then click **Submit**.
6. Click **Ok**.

Create labels

The basic steps for creating labels are:

1. Select **Labels**. All new installations default to Label 1, Label 2, Label 3.
2. Click the radio button next to the label name you want to edit, then click **Edit**.
3. Change the default name to an appropriate label name, then click **Save**.
4. Click the radio button next to the new label name, then click **View Items**.

   If no label items exist, the screen displays a message.
5. Click **Add Item**.
6. Enter the Item Name and click **Save**.

Create trigger systems

You can create trigger systems using the **Workspaces** tab. Creating the trigger systems is described in the *Detect User Guide*.

Set up feed files and perform any needed preprocessing

After the administrator has set up the environment and the business users have created the trigger systems, you need a set of transaction files to feed into the Detect Engine. Depending on the types of files that you need for your run, you might also need to run one or more of the Detect tools.

Configure cluster units and system settings

Before you run the system you need to configure the clusters and define the system settings using the Configuration Utility.

To run the engine

You can run the engine from the Engine Manager or from the command line.

Chapter 4. Creating Display Names for List items

As business users work with components, they select items from the drop-down lists in the Component Editor. These items represent "raw" values that come directly from your data sources. The Presentation Layer Manager allows you to assign business-friendly display names to these raw values. Display names make the component-building process more intuitive and efficient for business users.

For example, a high value customer who regularly interacts with your company might be categorized in your data source as *A+*, while a customer who interacts with your company infrequently is categorized as *I*. To help users understand these coded values easily, each logical name may be mapped to a display name, as follows.

Logical name	Display name
A+	Best Customer
I	Infrequent Customer

When a user accesses the Component Editor to build a component to target infrequent customers, the user sees the display name **Infrequent Customer**, not the logical name.

About the presentation layer list

The mappings for logical names to display names are held in a presentation layer list that specifies the display values for fields in a data source.

The Presentation Layer Manager allows the administrator to perform necessary list and list item maintenance tasks without directly accessing the database in which they are stored.

To create a new presentation layer list or replace the contents of an existing list, you can either import a comma separated value (CSV) file containing the list items, or enter the list items manually using the ***List Item Interface***.

You can delete or edit list item mappings, but first the system checks if components use the item and prevents the list item from being deleted or edited if it is in use. A popup message, displaying the component ID and workspace, indicates when components are affected. You must edit the affected components to not use the list item, or delete affected components, before you delete or edit the list item.

Creating presentation layer lists

The Presentation Layer Manager allows you to perform the following tasks.

- **Add List** — Create a new list, which is also known as a domain table.
- **Edit List** — Change the list name or description.
- **Delete List** — Delete an existing list and the associated list items.
- **Add List Item —** Creates a new item that is added to the specified list.
- **Edit List Item —** Change the logical or display name of an existing list item.

- **Delete List Item —** Delete a list item that is no longer needed.

To add a new list of items

Follow these steps to create a new list, either empty or with content, using input from a comma-separated value (CSV file). These lists are also known as domain tables.

1. Click **Presentation** in the navigation bar.

   The screen displays a tree structure of the existing lists on the left side of the page. The right side of the page will be empty until you choose an operation.
2. Click **Add List**.

   The right side of the page displays the fields associated with creating a list.
3. In the **List Name** text box, type a unique name for the new List.

   The name must begin with a letter and may contain only letters (upper or lower case), and the underscore character (_). The name can be a maximum of 50 characters.
4. In the **List Description** text box, type in a description for the list.

   The description can have a maximum of 200 characters. Descriptions can contain apostrophes or quotation marks.

   Descriptions can contain apostrophes or quotation marks.
5. If you want to create an empty list, select the radio button **Create Empty**.

   If you want to create a list with content (and create the list and the list items in a single step), select the radio button **Create with Content**, then click **Browse** and select the file that contains the contents of the list. The file must conform to the following constraints:
   - Name of the file must be in .csv format.
   - File must be a comma delimited file.
   - File must include a header row.
   - There cannot be any null fields.
   - No duplicates are allowed.
   - Maximum of 36 characters for logical and display names.
   - No trailing return at the end of the file.

   Here is an example of what the file should look like:

   LogicalName1, DisplayName1

   LogicalName2, DisplayName2
6. Click **Add**. A confirmation message appears when the List has been created successfully. If you entered a List Name that is not unique, the system will prompt you to enter a unique name.

To edit a list

Use these steps to change the name or description of an existing list

1. Click **Presentation** in the navigation bar to display the tree of existing lists.
2. Click on the name of the list that you want to edit.

   The right side of the page displays the fields associated with that list.
3. Update the editable fields as needed and then click **Save**.

To delete a list

Follow the instructions below to delete an existing list and the associated list items. The system will not allow you to delete a list that is used by trigger components.

1. Click **Presentation** in the navigation bar to display the tree of existing lists.
2. In the tree-structure display, click the name of the list to be deleted.
3. Click **Delete List**.

   If the list is in use by trigger systems it cannot be deleted. The system will notify you.
4. If items in the list are not in use, a pop-up message requires a response. Click **OK** to delete the List, or **Cancel** to cancel the delete.

To add a list item

Use these steps to add items to a list.

1. Click **Presentation** in the navigation bar to display the tree of existing lists.
2. On the left side of the page, click on the name of the list to which you want to add items.
3. Click the **Add List Item** button.

   The right side of the window prompts you to enter a logical and display name for an individual list item.
4. Enter both the **Logical Name** and the **Display Name** for the list item.

   The following guidelines apply to list item names:
   - Names must be unique.
   - The name must begin with a letter.

     The remainder of the name can contain letters, numbers, and the following special characters: parentheses, dollar sign, hash, underscore, plus sign, minus sign, comma, period, and space ($ # _ + - , .).
   - Names cannot exceed 36 characters.
5. Click **Add**.

   You will know that the new item was added when the it appears in the tree-structure display, as a child in the List and the screen changes to the Edit List Item editor. You can make changes to your newly added item.

To edit a list item

Use these instructions to change the logical name or the display name of a list item.

Note: The system will not allow you to edit list items that are used in trigger components.

1. Click **Presentation** in the navigation bar to display the tree of existing lists.
2. In the tree-structure display, click the parent-child button to expand the display and view the items associated with the list.
3. Click on the list item that you want to edit.
4. In the Edit List Item editor, enter the new logical name or display name.

   Note: If items in the list are in use by trigger systems they cannot be edited.
5. Click **Save**.

To delete a list item

Note: The system will not allow you to delete a list item that is used in trigger components.

Use these steps to delete an item from a list.

1. Click **Presentation** in the navigation bar.
2. In the tree-structure display, click the parent-child button for the list you want to edit so that you can see the items associated with the List.
3. Click on the list item that you want to delete.

   The right side of the window displays the details for that item.
4. Click **Delete List Item**.

   If items in the list are in use by trigger systems they cannot be deleted. A popup will notify you that it is in use.
5. A pop-up message requires a response. Click **OK** to delete the items, **Cancel** to cancel the delete.

Chapter 5. Defining Data Sources

A data source in IBM Unica Detect is an external source of information read by the Detect engine for processing against trigger systems.

There are four data source types:
- **Profile feed files—**These contain entity profile data in a flat file format.
- **Transaction feed files—**These contain raw transactions in a flat file format. At least one Transaction Feed File data source is required to run the Detect engine.
- **Lookup table—**This is an external database table that contains lookup data.
- **Profile table—**This is an external database table that contains entity profile data such as state of residency, age, gender.

Entity profile data is not required. However, when specified, it can be read from either a profile feed file, a database table, or a combination of both.

The Data Sources manager enables you to describe the data sources and to perform these tasks.
- Sort the list of data sources by clicking any of the underlined column titles
- View the details of a data source by clicking its data source name
- View and modify the fields defined for a data source by clicking its **Fields** icon
- Navigate to other pages in the data source list by clicking on the page number under the list of data sources

Note that you should have caching turned off in your browser when you work with data sources, to ensure that you see the latest changes. In Internet Explorer, select **Tools** > **Internet Options**. On the General tab, click **Settings** in the Browsing History section. Select **Every time I visit the webpage**.

About defining a data source in Detect

There are two ways you can define data sources within Detect.
- You can create a new data source.
- You can copy an existing data source.

You must define the fields that are contained in each data source that you set up. The data source fields identify the fields to use from the data source. The fields you define are the fields that are available to use when creating a component.

To define a new data source

Use these steps to create a new data source.
1. Click **Data Sources** in the navigation bar.

   The screen lists any existing data sources.
2. Click **Add**.

   The window expands so you can enter information about the data source. Use the **Data Source Details** section at the bottom of the page to define a new data source.
3. Select the **Source Type** from the drop-down list.

The source type describes both the format and the function of the data source. There are four data source types. The format of the source type is either a text file or database table.

The information that you need to provide in step 6 varies, depending on which source type you select in this step.

4. Use the information below to populate the **Logical Name**, **Display Name**, and **Description** fields.
   - **Logical Name** is the name used internally by Detect to recognize the data coming in on this field. This name must be unique within the database. Do not use any spaces.
   - **Display Name** is a user-friendly name for the name of the data contained in this data field. This is the name displayed in the component editors. It must be unique within the database. Do not use any leading spaces.
   - **Description** provides information about the data field. This is optional information.
5. In the **Source Name** field, enter the distinguishing string associated with a file of this type. Do not use any spaces. This name must be unique in combination with the entity type, which means that you can use the same name more than once as long as entity type is different.

   For transaction or profile feeds, the source name should be the text that appears in the <*sourcename* > entry of the feed file. For a database table, the source name is the name of the table, and will probably be the same as what was entered for the logical name.

   File names follow the pattern: `unica.<entity type>.<sourcename>.<YYYYMMDDhhmmss>`

   Note the following.
   - The prefix used can be any of these values: `unica`, `detect`, `insights`, `marketsoft`, `elity`. The example above uses `unica` as the prefix.
   - The entity type is always a single, lower-case letter in the English alphabet, even if multi-byte characters are used elsewhere in Detect.
   - The time portion of the datestamp is optional, but provides greater granularity.
6. The Source Type you selected in step 3 controls which of the following fields you see. Select the values for the fields you see.
   - **Entity Type Name –** Select the entity that you will track in the data source. If the entity you want is not listed, click the **Add Entity Type** link to add it.

     Note: If you add a new entity type, the system cannot process it until you associate it with a cluster.

     See “To define an entity type within the Data Sources manager” on page 28 for details on adding an entity type.
   - **Use Default Profile –** Select this option if you want the system to include default profile values for transactions that come in without profiles.
   - **Preload –** Select **None** if you did not preload any data into memory at runtime and **Full** if you did preload data into memory at runtime.
   - **Connection –** Select the name of the connection from the list, which includes user-defined connections and system connections (such as for rule, log, outcome, and visitor). You can re-use the system connections if your lookup tables share a database or schema with the system tables. If your lookup tables are in a different schema or database, define connections for them.

Connections are required for data sources that are defined as either Profile Tables or Lookup Tables. If the connection you need is not listed, click **Add Connection** to add it.

- **File Encoding –** Select an encoding for profile and transaction feed files. The format you specify in this field must match the format in which the feed file is saved.
- **File Date Format –** Select the date format for profile and transaction feed files. The date format you specify in this field must match the date format used in the feed file.
- **File Currency Format –** Select the currency format for profile and transaction feed files. The currency format you specify in this field must match the currency format used in the feed file.

7. Click **Save** to add the data source to the list. You may have to page through the list to find the newly-added data source.
8. Optionally, you can click the **Fields** icon to define fields for the data source.

To create field definitions for a data source

Use this section to define the fields in a data source.

1. Click **Data Sources** in the navigation bar to display the Data Sources screen.
2. Find the name of the data source to which you want to add fields and click the **Fields** icon for the data source.

   The list of existing fields (if any) in the data source appears.
3. Click **Add** to display the Data Source Field Details window.
4. Type in the following information:
   - **Logical Name** (required) is the name used by Detect to recognize the data source field. Do not use any leading spaces. The logical name for the field must be unique within the data source.
   - **Display Name** (required) is the user-friendly name for the data source field. This is the name displayed in the component editors. The display name for the field must be unique within the data source. Do not use any leading spaces.
   - **Description** provides information about the data source field. This is optional information.
5. Select the **Field App Type** value.

   Note: The values that are available depend on the type of data source to which you are adding the fields. Likewise, the options in the remainder of the drop-down lists depend on the other selections you make.
6. Select the **Data Type** from the drop-down list.

   Data Type defines the type of data the field contains.
7. Select the **Display Type** from the drop-down list.

   The display type controls how users can enter information for the field within the Detect editors. For example, if the display type is Text, the field will have a text box. If the display type is List, then there will be a drop-down list from which to choose.
8. If the Display Type is **List**, then you must select a Domain Table that contains the values that are allowed for the field. Domain tables are lists defined in the Presentation Manager.

   Note: If you are working with a field in a data source that is already in use, you cannot change the **Domain Table Name** selection.

9. Click **Save** to add the field the list. You may have to page through the list to find the newly-added field.
10. Close the window.

About copying an existing data source

When you copy a data source:

- All of the fields are copied, with the exact same names and types as in the original data source.
- The copy has the same source type as the original data source; you cannot change the source type of the copy.
- You can change all other properties of the copy data source, including entity type.

To copy an existing data source

Use these steps to copy an existing data source. When you create a copy of a data source, you can edit it as if you were creating a new data source because it is not in use yet.

1. Click **Data Sources** in the navigation bar.

   The screen displays any existing data sources.
2. Click the name of the data source that you want to copy.
3. When the Data Source Details appear, click the **Copy**link.

   Because many of the names must be unique, they are automatically prefixed with **Copy_of_**.
4. Optionally change the properties that are editable.
5. Optionally edit the **Description** field to describe the data source you are creating.
6. Optionally select a new entity type using the **Entity Type Name** drop-down list. If the entity type you need is not listed, click the **Add Entity Type** link to create it.
7. Click **Save** to add the data source to the list. You may have to page through the list to find the newly-added data source.
8. Optionally, you can add, delete, or change fields in the data source definition.

About editing a data source

At any time, you can change the logical name, display name, source name, or description of a data source. Also, if the data source has a connection, you can change that at any time.

After you create a data source, you cannot change its source type. If the data source is in use, you cannot change its entity type.

If the data source has the **Is Default Profile** option selected, you can clear it at any time. However, if the IS_DEFAULT_PROFILE field is in use, the system will not remove it.

About deleting a data source

You can delete a data source if it is not being used by any trigger system. Detect automatically deletes the field definitions associated with it so that they are not orphaned within the system.

About changing a field definition for a data source

You can edit any of the Data Source Field Details settings if the field is not being used by any trigger system.

Note: If the field is in use, the following field properties cannot be changed: Field App Type, Data Type, and Display Type. If the display type is List, then the Domain Table Name property cannot be changed.

About a deleting a field from a data source

You can delete a field from a data source if it is not being used in any trigger system.

To edit a data source

1. Click **Data Sources** in the navigation bar.

   The screen displays any existing data sources.
2. Click the name for the data source that you want to edit.
3. When the **Data Source Details** section appears, click the **Edit** link.
4. Make your changes and click **Save**.
5. Optionally, you can add, delete, or change fields in the data source definition.

To change field definitions in a data source

Use this section to edit fields in a data source.

You can change the logical name, display name, and description of a data source at any time because the system's logic is based on the data source's unique ID, not on these fields.

1. Click **Data Sources** in the navigation bar to display the Data Sources screen.
2. Find the name of the data source from which you want to change fields and click the **Fields** icon associated with that data source.

   The list of existing fields in the data source appears.
3. Click the field name for the field you want to edit.
4. Click **Edit**.
5. Make the changes you want to make.
6. Click Save.
7. Click **Close** to close the window.

To delete fields from a data source

Use this section to delete fields from a data source.

Note: Fields cannot be deleted from a data source if they are in use.

1. Click **Data Sources** in the navigation bar to display the Data Sources screen.
2. Find the name of the data source from which you want to delete fields and click the **Fields** icon associated with that data source.

   The list of existing fields in the data source appears.
3. Click the field name for the field you want to delete.
4. Click **Delete**.
5. Click **Close**.

About entity types

The entity type categorizes a group of transaction data sources (and optionally profile data sources), identified by a common key. It serves as the reference point, used by the Feeder for grouping data sources, and by the Engine for maintaining the knowledge of events that have occurred for the defined trigger systems, and ultimately as the type of key for whom an action trigger has fired.

An entity type key is a single, lower-case character code, such as *c* for customer, that defines the characteristic of a data source.

How Detect uses entity types

The Feeder groups all data sources by their associated entity type. The Engine then recognizes events and maintains pattern state (the knowledge of events over time) by entity type. For example, the history of three deposits made to a checking account within a week is remembered by account number when the transactions are included in the account entity data feeds. History of the same three deposits will be remembered for a given customer when the transactions are included in the customer entity data feeds.

How you define entity types

There are two ways to define an entity type.

- You can define an entity type when you define a data source or edit a data source that is not in use, using the Data Sources manager.
- You can an define entity type using the Configuration Utility.

Examples of entity types

For example, in a financial implementation it is often necessary to categorize the same, or similar, day-to-day transactions by account number, customer identifier, and household identifier. In this case, you would create three entity types:

- **Account entity** – Account-related data source feeds are grouped within the account entity, by their common key, most likely an account number.
- **Customer entity** – Customer-related data source feeds are grouped within the customer entity, by their common key, perhaps the customer's social security number.
- **Household entity** – Household-related feeds are grouped within the household entity, by their common key, some unique household identifier.

Note: IBM Unica Services can advise you on the entities you need to define for your system.

To define an entity type within the Data Sources manager

You can define an entity type within the Data Sources manager when you are creating a new data source or copying an existing one. You can also define an entity type by editing an existing Data Source that is not in use.

1. Add a new data source.
2. Click the **Add Entity Type** link.
3. Enter information in the fields.
   - **Entity Type Name** – Give the entity type a descriptive name.

   - **Entity Type Code –**Give the entity type a single, lower-case alphabetic character code (using the English alphabet, even if you are using multi-byte characters elswhere in Detect). The code should reflect the purpose of a data source. For example, you might create *c* for customer.
4. Click **Save** to add the entity type to the drop-down list.

   Note: Before the system can process the entity type, you must associate the entity type with a cluster that will run it.

About connections to the data

The system requires connections for data sources defined as database tables, including lookup tables and profile data stored in tables. The connection provides the path that the engine uses to locate the database that contains the external table. Once you define a connection string, you can associate the connection with one or more data sources.

There are two ways to create connections to the data.
- You can define a connection when you define a data source in the Data Sources manager.
- You can define connections using the Configuration Utility.

To create a connection using the Data Sources manager

You can define an connection within the Data Sources manager if you are creating a new data source that is defined as database table, or copying an existing one. You can also define a connection by editing an existing database table data source.
1. Add a new data source that has a database table type or click on an existing one to edit it.
2. Click the **Add Connection** link.
3. Enter information in the fields.
   - **Connection Name –** The name used internally by Detect.
   - **Connection Description –** A user-friendly, descriptive name for the connection.
   - **Provider –** Select the provider of the database from the drop-down list.
   - **Server –** The name of the physical database server. This information is required only when the provider is set to SQLServer.
   - **Database Name –** The name that was given to the database service.
   - **User Name –** The user name that is used to log onto the database.
   - **Password** and **Re-enter Password –** The password associated with the supplied **User Name**. The password must be re-entered to detect mistakes.
4. Click **Save** to add the connection to the drop-down list.

About default profiles

To handle transactions that come in for accounts without profiles, you may optionally choose to have the system use default profile values. The system can run using default profile records, and trigger systems can be defined to filter them.

About the default profile field

When **Use Default Profile** is selected for a profile feed file data source, the system automatically adds a field called **IsDefaultProfile** (logical name IS_DEFAULT_PROFILE) to the profile data source. If the field is not in use, clearing the checkbox removes the field.

The field is a system-generated boolean (true or false) field and cannot be modified by users. When the field is included, values for it in the profile feeds are ignored by the Feeder. However, this field is available in the component editors so that it can be used for building trigger logic.

About using the IS_DEFAULT_PROFILE field when building triggers

Users building trigger systems can test for the IS_DEFAULT_PROFILE field's value to filter out default profile records. For example, they could build components that check the value of the field and only fire when it is false. The ability to test for this value is useful when building forward-looking inactivity trigger systems.

About default profile values

The default profile files contain system-defined dummy values:
- String field: `""` (empty string)
- Number: `0`
- Money field: `0.0`
- Boolean field: `0` (false)
- Date field: `1899-12-30 12:00:00 AM`

To enable default profiles

You can enable default profiles for profile feed files by selecting the Use Default Profile checkbox for the data source.

List of fields available for each data source type

The following tables identify the options that are available when you create data source fields.

Table 5. Transaction Feed File data source type

Field application type	**Data type(s) and display**	**Description**
Entity ID	string Displays as text.	The Entity ID indicates that the field is the unique record identifier. One field must be assigned the Entity Id Field App Type.

Table 5. Transaction Feed File data source type (continued)

Field application type	Data type(s) and display	Description
Carry On Value	Data types: • number Displays as text or list. • money Displays as text or list.	The Carry On Value indicates that the value of this field is used in a pattern expression evaluation when a pattern component includes a pattern expression. Only one field, per data source, can be defined as the Carry On Value. If no field is designated as the carry on, pattern components that include a pattern expression will produce unexpected results.
Transaction Attribute	Data types: • string Displays as text or list. Text is displayed as a text box in editors. List is displayed as a drop-down list in editors. • number Displays as text or list. • date Displays as text, list, or date. List can be list of holidays. Date is displayed as a text box. • boolean Displays as a list. Only from a yes/no list. • money Displays as text or list.	The Transaction Attribute is any other field that describes the transaction.
Transaction Date	date Displays as text, list, or date.	The Transaction Date indicates the date of the transaction.

Table 6. Profile Feed File data source type

Field application type	Data type(s) and display	Description
Entity ID	string Displays as text.	The Entity ID indicates that the field is the unique record identifier. One field must be assigned the Entity Id Field App Type. This should be a unique profile identifier, such as an account number or customer ID.

Table 6. Profile Feed File data source type (continued)

Field application type	Data type(s) and display	Description
Profile Attribute	Data types: • string Displays as text or list. • number Displays as text or list. • date Displays as text, list, or date. • boolean Displays as a list. • money Displays as text or list.	The Profile Attribute is any other field that describes the profile. It indicates that the field contains data on which components can be written.

Table 7. Database lookup table data source type

Field application type	Data type(s) and display	Description
Entity ID	string Displays as text.	The Entity ID indicates that the field is the unique record identifier. One field must be assigned the Entity Id Field App Type.
Profile Attribute	Data types: • string Displays as text or list. • number Displays as text or list. • date Displays as text, list, or date. • boolean Displays as a list. • money Displays as text or list.	The Profile Attibute indicates that the field contains data on which components can be written.
Carry On Value	Data types: • number Displays as text or list. • money Displays as text or list.	The Carry On Value indicates that the value of this field is used in a pattern expression evaluation when a pattern component includes a pattern expression. Only one field, per data source, can be defined as the Carry On Value. If you use pattern components that include a pattern expression, you must designate a field as the carry on. Otherwise, the pattern component will produce unexpected results.

Table 8. Profile table data source type

Field application type	**Data type(s) and display**	**Description**
Profile Attribute	Data types: • string Displays as text or list. • number Displays as text or list. • date Displays as text, list, or date. • boolean Displays as a list. • money Displays as text or list.	The Profile Attribute is any other field that describes the profile. It indicates that the field contains data on which components can be written.
Entity ID	Data types: • string Displays as text. • number Displays as text or list. • money Displays as text or list.	The Entity ID indicates that the field is the unique record identifier. One field must be assigned the Entity Id Field App Type.

Chapter 6. Managing Users and Security

Starting with the 8.2.0 release, IBM Unica Detect provides roles and permissions to control users' ability to view or modify Detect resources.

You administer Detect users and roles on the User and User Roles & Permissions pages. To access these pages, you must log in with a Detect account that hais authorized to access User Administration. By default, the pre-defined **system** account, which has an initial password of **host**, has this access.

You set password policies through the Configuration Utility.

Managing users

You must create a Detect user account for each person who requires access to Detect.

You manage Detect user accounts on the following pages.
- The Users page, where you can create and disable user accounts and manage roles for individual users.
- The User Roles & Permissions page, where you can assign roles to users and remove them, and create and modify roles.

You can also track user activities on the following pages.
- The User Login Activities page, which lists login and logout actions.
- User Audit Trail, which lists changes to user accounts.

About user name requirements

Detect user names must meet the following requirements.
- User names are limited to 50 alpha-numeric characters.
- User names cannot contain spaces.
- User names must be unique; no duplicate user names are allowed.

User names are not case-sensitive.

To create a user account

1. Log in to Detect as a user with appropriate permissions.
2. Select **Settings > Users** to go to the Users page.
3. Click the **Add a new user** icon.
4. Complete the fields and click **Save Changes**.

To delete a user account

When you delete a user account, it remains in the Detect database, and you cannot create another account with the same user name. However, the account is archived and no longer allows access to Detect.
1. Log in to Detect as a user with appropriate permissions.
2. Select **Settings > Users** to go to the Users page.

3. Click the username of the account you want to delete to open the user's page, and click **Delete User**.

   A Delete Confirmation section appears.
4. Enter the reason for deleting the account in the **Remarks** field and click **Save Changes**.

   Your remarks appear in the User Audit Trail page.

   The user name is removed from the Users page, and no one can log in to Detect using that account's credentials.
5. Click **Save Changes**.

To disable or re-activate a user account

When you disable a user account, it remains in the list of users and retains its assigned security roles. You can activate the account again at any time.

1. Log in to Detect as a user with appropriate permissions.
2. Select **Settings > Users** to go to the Users page.
3. Click the username of the account you want to disable to open the user's page.
4. Click **Edit Properties** and do one of the following in the **Status** field.
   - To disable the account, select **Disabled**.
   - To re-activate the account, select **Active**.
5. Enter the reason for disabling or re-activating the account in the **Remarks** field and click **Save Changes**.

   Your remarks appear in the User Audit Trail page.

   The user name remains on the Users page, but no one can log in to Detect using that account's credentials.
6. Click **Save Changes**.

To reset a Detect user's password

1. Log in to Detect as a user with appropriate permissions.
2. Select **Settings > Users** to go to the Users page.
3. Click the user name of the account to open the user's page, and click **Reset Password**.
4. Enter the new password twice, and enter any remarks you want to appear on the User Audit Trail page.
5. Click **Save Changes**.

   Note: Any password policies that are set for the system are enforced. If the password does not comply with the system's password policies, an error message appears.
6. Inform the user the of new password.

About password policies

Password policies are optional. A Detect user who has permission to modify password policies uses the Configuration Utility to turn the enforcement of password policies on or off and set the policy values. By default, the pre-defined **system** account has this access.

If password polices are turned on, they are enforced each time a user logs in to any part of the system, including the user interface and all tools that require a log in.

Passwords are stored in an encrypted format.

Note: For new installations, password policies are turned on by default. For installations that are upgraded from a version earlier than 7.2.2, password policies are turned off by default.

About Detect tools that require login

The following Detect tools require a log in.

- Configuration Utility
- Outcome Management Tool (OMT)
- EDM Driver
- Entity State Optimizer (ESO)

Note: If password policies are turned on and the system fails to run, first check the EDM Driver logs to determine if a required password has expired. If the password has expired, try to log in to Detect using the expired password. Detect will allow you to change the password.

To set password policies

Password policies are optional and can be configured in the Configuration Utility by the Detect system administrator.

1. Run `ConfigurationUtility.exe`, located in the `Application\bin` directory under your Detect installation, and and log in using an account with appropriate permissions.

   You must have the appropriate permission to run the utility (for example, you can log in as the **system** user or any user with a role that includes the **Password Policy** permission).

   If your permissions do not allow you full access to the Configuration Utility you will not see a tree or multiple tabs.
2. On the System Configuration tab, adjust the password policy settings.

   See the "Password policy settings in the Configuration Utility" for further details about each option and their default values.

   You must set **Apply User Password Policies** to **YES**, or none of the other options will take effect.
3. Click **Apply**.

Password policy settings in the Configuration Utility

Setting	Description	Default
Apply User Password Policies	If this value is set to YES, the system checks password policies each time a user password is set and when a user is authenticated. If this value is set to NO, no policy is applied during either activity. **Note:** This setting acts as a switch for the password policy settings. Password policies are only enforced if this item is set to YES.	YES for new installations, NO for upgraded installations.
Force to Change Password at First Login	If this value is set to YES, the system forces Detect users to change their passwords the first time they log into the system as new users or after the system administrator has reset their passwords. If this value is set to NO, users are not required to change their passwords in these cases.	YES

Setting	Description	Default
Minimum Length	The minimum length of a password.	8
Minimum Number of ASCII Alphabetic Characters	The minimum number of ASCII alphabetic characters that must be present in a password. **Note:** Passwords can also contain Latin-1 characters, but this setting specifies a required minimum number of ASCII alphabetic characters.	5
Minimum Number of Numeric Characters	The minimum number of numeric characters that must be present in a password.	1
Number of Days Until Password Expires	The number of days a user's password can exist before it expires. Value of 0 means the password never expires. **Note:** The password for the Detect**system** account never expires.	60
Number of Failed Login Attempts Allowed	This value sets number of failed login attempts allowed. The user account will be locked after the value is reached. Value of 0 means that user is never locked regardless how many times failed login occurs.	4
Password History Setting	This value sets the number of unique new passwords required before an old password can be reused. For example, if the number is 5 then the system keeps track of each user's past 5 passwords. When users change their passwords, they cannot re-use any of their past 5 passwords. Value of 0 means that there is no restriction.	5
Password Can Be Same As User Login Name	If set to YES, users can set their password to be the same as their user login names.	NO

Managing roles and permissions

A permission is an action that a user can perform on a resource (a data object or a web page). A role is a set of permissions.

Generally, you should give users roles with permissions that reflect the functions that users perform in your organization.

You can use the pre-defined roles described in "Pre-configured user and roles" on page 40 and you can also create custom roles that reflect your organization's security requirements.

About creating custom roles

The Modify permission includes the View permission, so you do not need to grant both if you want a role to allow both view and modify access to a Detect resource. You should grant only the View permission if you want to restrict access to read-only.

Roles are evaluated in a cumulative fashion. For example, suppose Role A allows only View access to the Data Sources page, and Role B allows Modify access to the Data Sources page. A user with only Role A would have read-only access to the Data Sources page, but a user with both Role A and Role B would be able to view the Data Sources page and perform all available actions there.

Access control

Roles control access to each of the following web pages and utilities.

Web pages

- Workspace
- Runset
- Data Source
- Presentation
- Type Descriptor
- Label
- Engine Manager
- Reports
- Response
- User Administration (includes the Users, Roles & Permissions, User Audit Trail, and User Login Activities pages)

System Utilities

- Configuration Utility
- Password Policy (a function within the Configuration Utility)
- Start Transaction Run
- Outcome Manager
- Entity State Optimizer
- Ramp Up
- Events Pre-Processor
- Feed Validator
- Response Manager

Permission states

The following states apply to permissions for Detect web pages.

- **View—**Applies to all Detect web pages. Allows the user to view the page but not perform any actions.
- **Modify—**Applies to all Detect web pages except Engine Manager and Reports. Allows the user to view the page and perform all available actions on the page, including create, edit, delete, copy, and publish.
- **Run—**Applies only to Engine Manager. Allows the user to initiate a run.
- **Audit—**Applies only to user administration. Allows view access to the User Login Activities and User Audit Trail pages.

For the Detect system utilities, you can grant or deny access to the utility as a whole, except for the Configuration Utility, where you can separately grant or deny access to the Password Policy function.

Pre-configured user and roles

Detect provides one pre-configured user account and four pre-configured roles.

About the system user account and the Host role

Detect has one built-in user account, named **system**, with an initial password of **host**.

Important: You should change the password of the **system** account and do not retain the default password, as all installations of Detect use the same initial password for the **system** account.

The **system** account is assigned the **host** role, which allows the following access.
- In the Detect web application, it allows full access only to the user administration pages
- Among the system utilities, it allows access only to the password policy function of the Configuration Utility

To ensure that an account with user administration privileges always exists in Detect, the following restrictions apply to the **system** account and the **host** role.
- You cannot modify or delete the **host** role or assign to any other user.
- You cannot delete or disable the **system** account.
- You cannot assign any additional roles to the **system** account, but you can change the password.
- The **system** user is not visible on the Users page unless you log in using the **system** account.
- The **host** role is not visible on the User Roles & Permissions page unless you log in using the **system** account.

You should create a comparable user account and role for everyday use, and reserve the **system** account and the **host** role for use if you need to restore access to Detect in case the everyday user administration account is accidentally deleted or disabled. If you lose access to the user administration features of the Detect web application because all accounts with this permission are deleted or disabled, contact IBM Unica Technical Support.

About the pre-configured roles that you can modify

The following roles are provided by default. You can modify the pre-configured roles to meet your organization's security requirements.
- **Default** is automatically assigned to every new user. It allows view access to all Detect web pages except user administration. This role cannot be deleted, but you can modify its permissions to meet your organization's needs.
- **Power User** is assigned to every existing user when you upgrade. It allows all views and actions on Detect web pages except user administration, and access to all system utility functions except setting password policy.

To create a role

1. Log in to Detect as user with appropriate permissions.
2. Select **Settings > User Roles & Permissions** to go to the User Roles & Permissions page.
3. Click the **Add new role** icon.

4. Complete the fields to give the role a name and meaningful description, and click **Save Changes**.

   The name must not be the same as any other role in the system.
5. Click **Edit Permissions** and select the desired combination of permissions for the role.

   You can use the **Select All** and **Deselect All** buttons to speed the process when you want to assign either many or few permissions.
6. Click **Save Changes**.

   The new role is listed and is available for assignment to users.

To assign or remove a role

To assign a role to a user account or to remove one, you must log in to Detect a user with appropriate permissions.

You can assign and remove roles in either of the following ways.

- On an individual user's page, by clicking **Assign Roles** and moving roles into and out of the **Assigned Roles** box.
- On the role's page, by clicking **Assign Users** and moving users into and out of the **Assigned Users** box.

To modify role permissions

1. Log in to Detect as user with appropriate permissions.
2. Select **Settings > User Roles & Permissions** to go to the User Roles & Permissions page.
3. Click the role you want to modify to open the role's page.
4. Click **Edit Permissions** and modify permission selections as desired.

   You can use the **Select All** and **Deselect All** buttons to speed the process when you want to assign either many or few permissions.
5. Click **Save Changes**.

To delete a role

1. Log in to Detect as user with appropriate permissions.
2. Select **Settings > User Roles & Permissions** to go to the User Roles & Permissions page.
3. Click the name of the role you want to delete and click **Delete Role**, then click **OK** in the confirmation message.

   The role is deleted.

About the User Audit Trail and User Login Activities pages

A user with appropriate permissions can access the User Audit Trail and User Login Activities pages. These pages provide the following information about user activities in the system.

- User Audit Trail—Tracks when users are created, modified, archived, disabled, have their password reset, and are assigned to or removed from a role. A timestamp is provided, along with the user name of the user who performed the action and any remarks entered by that user.
- User Login Activities—Tracks each login and logout and whether the action occurred in the web application or a utility. If the action occurred in a utility, the

utility is named. A timestamp is provided, along with the user name of the user who performed the action and, if the action failed, the reason for the failure.

Chapter 7. Creating Type Descriptors

Type descriptors inIBM Unica Detect define the characteristics, format, and layout of data used in Detect. The type descriptor acts as a framework for the data, defining column names, the type of data for the column, whether the column allows a null value, whether the column is used as a key, and so on.

After you create a type descriptor, you can select it as the data type for data that can be stored in a Container, or manipulated by the Select and/or Join functions.

In a Container, the data type definition is associated with the data stored in the Container. In a Select or Join function, the data type definition is associated with the result of the function.

You use the Type Descriptors manager to create, edit, copy, and delete type descriptors.

To create a new type descriptor

1. Select the **Type Descriptors** in the navigation bar to display a list of existing type descriptors.
2. Select **Add New** in the navigation tab area.

   The screen updates to display the type descriptor fields.

   If the type descriptor will be used in a container component, check the **Use for Container** checkbox at the top of the screen.

   The screen will refresh to show one additional field (**Compression**) in the **Field Attributes** area. In addition to the fields and checkboxes, a date field (**TimeStamp**) is automatically included in the type descriptor.
3. In the **Type Descriptor Name** text box, enter the name.
4. Define the **Field Attributes** for each field in the type descriptor. The **Field Attributes** describe the characteristics of a specific column in the type descriptor you are creating. The order of the field columns within the type descriptor is determined by the order in which you add fields.
   a. In the **Field Name** text box enter the field name.
   b. Select a **Data Type** from the drop-down list. The types are:
      - **Int32**
      - **Double**
      - **DateTime**
      - **Boolean**
      - **String**
   c. If the field is defined as a **String** data type, enter the **Size** of the field.
   d. For a type descriptor that is used for a Container, select the Compression parameter.

      Depending on the data type, you may see different options. The Compression types include:

Compression Parameter type	Description
Last	Enables you to explicitly add a value to a container that is being compressed. For example, if the container is compressed daily, and you want to hold the amount (compressed by Sum), count (compressed by Sum), and daily average (explicitly set by a Math expression), then use the Last operator when setting the compression behavior for the daily average field in the type descriptor.
Sum	When compression takes place, add together the values in this field. This option is available for data types of Int32, Double, DateTime.
Group By	When compression take place, use this field to group the same values together. This option is available for all data types.
Or	Available only for a Boolean data type, performs an or operation on the values in the field. True condition results when all the values in the field are true, or when there is a combination of true and false values. False condition results when all the values in the field are false
Blank	Not used in the compression. This option is a available for all data types

e. Select a **Field Type** from the drop-down list. The options you see may depend on the other selections you have made. The types are:
   - **As Attribute —** This option is available for all data types; it is the most common setting. Use it for a field you might just want to look at. If this type descriptor is for container use, this is the only option you will see.
   - **As Date —** This option enables date data to be used for calculations.
   - **As ID —** While not typically used, this setting designates the field as a unique identifier for a join.

f. Enter a **Default Value** for the field if one is required.

g. Check the following checkboxes as applicable:
   - **Allow NULL —** when checked, the field can contain a NULL value.
   - **Is Unique —** when checked, the system will only insert the first, unique instance of the field into the container.

     If the type descriptor is used on a container, then the field set as unique acts a type of unique key. If a new set of values is to be inserted into the container and the value for the unique field is the same as a value that is already in the container (in that field), then that entire row will not be inserted.

     For example, it could be used to store the date of the first time a customer purchased some item.You could define a container with a date field and a product ID number, and set the product ID as a unique field. When a customer buys item 001, the ID 001 and the date of the purchase will be stored in the container. When the same customer buys item 001 again, the new date will not be inserted because 001 already exists in the container in the unique field.

h. Check the appropriate radio button: **Not Key** or **Primary Key**.

   Note: These parameters are not offered when a type descriptor is used for a Container.

    i. Click the **Add Field** button on the right-hand side of the page.

       As you add fields to the type descriptor, the editor displays their attributes in the lower portion of the screen.

5. Continue defining and adding all the fields, as described above in step 4.
6. Click **Save Definition** when complete. Or, if you want to delete all the fields you have been defining, click **Clear Definition**.

To edit fields in a type descriptor

Be carfeul about changing the field type of an existing type descriptor. Normally, you should change the field type only during the process of developing and testing trigger systems. After a field type is used in a component that is part of a production trigger system, you should not change the field. type.

1. Select the **Type Descriptors** in the navigation bar to display a list of existing type descriptors.
2. Click the checkbox associated with the type descriptor to be edited and then click **Edit**.

   The page to edit the type descriptor appears. The page shows different information depending on whether the type descriptor is in use or not.

   If the type descriptor is in use, the page lists all rules that use the type descriptor. The buttons to **Add Fields**, **Remove Fields**, and **Clear Definitions** are disabled so that users cannot add, remove, or clear fields. The **Edit** link for each field is enabled so that use can still change specific field definition and click **Save Definition**.
3. In the table area where the fields are listed, click the **Edit** link associated with the field to be modified.

   The table row will refresh so the values can be edited and the Edit link will be replaced with an **Update** and a **Cancel** link.
4. Modify the field attributes as needed.
5. Click **Update** associated with the field being modified.
6. Click **Save Definition** (in the Descriptor Controls area of the screen).

To delete a field from a type descriptor

1. Select the **Type Descriptors** in the navigation bar to display a list of existing type descriptors.
2. Click the checkbox associated with the type descriptor to be edited and then click **Edit**.
3. In the table area where the type descriptor fields are listed, click the Remove checkbox that is associated with the field to be deleted.
4. Click **Remove Field** (in the Descriptor Controls area of the screen).
5. Click **Save Definition** to save the changes to the type descriptor.

To copy a type descriptor

1. Click the checkbox associated with the type descriptor to be copied and then click **Copy**.

   The page to edit the type descriptor displays. All items in this window, even the name, are identical to the type descriptor you copied.
2. In the **Type Descriptor Name** textbox enter the name for the new type descriptor.

Type descriptors must have unique names.

3. Add fields, edit fields, and remove fields as desired to create the new type descriptor.
4. In the Descriptor Controls area of the screen, click **Save Definition**.

To delete a type descriptor

You can delete type descriptors, as long as they are not being used. The system will not allow you to delete one that is in use.

1. Select the **Type Descriptors** in the navigation bar on the Type Descriptor Manager screen.

   A list of existing type descriptors is displayed.
2. Click the checkbox associated with the type descriptor to be deleted and then click **Delete**.
3. When prompted to continue with the deletion, click **OK** to delete the type descriptor.

Chapter 8. Creating Sort Categories for Components Using Labels

Component **labels** in IBM Unica Detect provide a way to categorize components for sorting and organization purposes. You create these components in the Label Manager.

Three label categories are available for each component, and you can add label items it each of these top-level categories. The default names for the label categories are **Label 1**, **Label 2**, and **Label 3**. You assign meaningful names using the the Label Manager.

You must create **Label Items** for each label category. Label Items are the values associated with each of the three label categories. When the component owner creates a component, he or she can assign up to three of these label items to each component, one for each label category. See the *Detect User Guide* for more information.

Label categories and label items appear on the Component List tab in the Workspace Editor. For example, Label 1, Label 2, and Label 3 could be named **Trigger Number**, **Trigger Name**, and **Category**. Clicking the label name sorts the components alphabetically by label item. Thus, in the example, you could sort components according to the name or number of the trigger where they are used.

When you change the name of a label, it does not change the associated label items.

When you change a label item, the change affects all components associated with that label item. To change a label item for one component and have the change affect only that component, you must do the following.
- Add a new item to the label.
- Edit the component in the Component Manager to replace the old label item with the new label item.

To modify Component Label Names

1. Select **Labels** in the navigation bar to display the Label Manager.
2. Click the radio button associated with the label name to be modified and then click **Edit**.
3. Enter the new label name and click **Save**.

   Label names can have a maximum length of 25 characters and must begin with an alpha-numeric character.

To manage Component Label Items

You can add, modify, or delete component label items; all of those processes start with accessing the Label Manager.
1. Access the Label Manager by selecting **Labels** in the navigation bar.
2. Either click on the label name or select the radio button associated with the label and click **View Items**.

The screen updates to show the items associated with the selected label.

3. Continue to one of the related sections, depending on what you want to do next.

To add Component Label Items

1. While viewing the label items, click **Add Item** to display the New Label Item popup.
2. Enter the new item name and click **Save**.

To modify Component Label Items

1. While viewing the label items, either click on the label item name or select the radio button associated with the label item and click **Edit Item**.
2. Edit the item name and click **Save** to complete the modification.

To delete Component Label Items

1. While viewing the label items, select the radio button associated with the label item that you want to delete and click **Delete Item**.
2. When you are prompted the second time, click **OK** to complete the deletion.

Chapter 9. Managing Workspace Libraries

If you use any of the workspaces available with an Detect trigger library available for IBM Unica Detect, or if you create your own library of workspaces, you may need to export or import all or some of the workspaces in these libraries using the Detect library management tools.

You can use the Detect library management tools to update your library with new workspaces, or to streamline the workspaces used by the Detect engine while maintaining the other workspaces in a file that you store outside of your Detect system. You can also use the library management tools to maintain backups of our trigger system libraries within your databases.

Note: Before you can use the library management tools, you must perform some preliminary configuration steps.

About library management tools

Detect provides four library management tools that can assist you in exporting, importing and comparing of Detect trigger systems from one environment to another. Exports include the workspace and all its rules and can optionally include all the dependent data sources, type descriptors, entity types, presentation lists, and labels. The four tools are:

- **ExportManager –** ExportManager is a command-line tool that exports a system-independent, .xml representation of a specified trigger system. Basic options allow users to include one or more workspaces, to include the dependencies for these workspaces, and to limit the exported dependencies to only those needed by the rules in the specified workspaces.
- **DiffManager –** DiffManager is a command-line and/or graphical tool that allows users to see the differences (prior to import) between the rules in an export file, and those in a Detect database (into which a user would subsequently import). The command-line interface can generate a series of text files that describe the differences it finds.
- **ImportManager –** ImportManager is a command-line tool that imports a previously exported .xml representation of a trigger system, including any dependencies contained within the specified import file.
- **LibraryManager –** LibraryManager is a graphical interface for the ExportManager, ImportManager, and DiffManager command-line tools. Using the graphical interface, users can connect to a specified database, import a selected file, or export a selection of workspaces.

About using library management tools for database management

If you maintain two copies of the same database, or portions of a database, and you make a change to one of those copies, you must make the same change in the other copy in order to see the same behavior in Detect. If you manually make identical changes to the databases using Detect, the two databases (or portions of the databases) are no longer considered to be identical, because every component does not have the same ID. Instead of making the changes manually, you can use the library management tools.

How the tools identify matching objects

When you use the Library Manager to import from one environment to another, it checks to see if the objects you are moving already exist within the destination environment. Most objects are matched on their uniqueID, which is a unique, system-generated identifier.

However, the following three objects have additional attributes that must be unique within the environment:

- workspace (unique runspace code)
- entity type (unique entity type code)
- type descriptor (unique name)

Therefore, the Library Manager will match those objects based on these alternative attributes if it does not find a match based on unique ID. For example, if an entity type already exists that has the entity type code a, then the Library Manager matches the two entity types. It then changes the uniqueID of the object in the target environment so that the uniqueIDs match. Type descriptors and workspaces are treated in a similar manner. All other objects are matched based on uniqueID only. UniqueIDs match across environments only if the object has already been imported from one environment to the other (or if a database was backed up and restored between environments).

About setting import and export options

You can set import and export options within the Library Manager, so that you do not have to enter them each time.

To perform database management using the tools

If you have the need to maintain more than one copy of a portion of your database, you can use the library management tools to streamline the database management for Detect as follows:

1. Make any necessary changes to the database you intend to export.
2. Export the changed portion of the database you want to copy, from the Library Manager or the command line.
3. Import the portion of the database you exported in Step 2 to the other database, from the Library Manager or the command line.

When you follow this procedure, any changes you make to a database in Step 1 are identically copied into the database you use for import in Step 3. Consequently, the two portions of the databases are identical and you can efficiently maintain them for use by Detect.

To prepare to use library management tools

Before you can use the Detect library management tools, you must complete the instructions in the following sections.

To unzip the library management tools

1. Locate the `librarymanager.zip` file within your product media.
2. Unzip the `librarymanager.zip` file to any location on your computer. Preserve the directory structure when you unzip this file.

After you unzip this file, the top directory for this tool is called librarymanager. Several directories are automatically created beneath `librarymanager`, including `/bin`, `/conf`, and `/lib`.

To copy the jdbc driver for your database to the lib directory

1. Locate the jdbc driver for your database type.
   - **Oracle**: For Oracle, use `ojdbc14.jar`. For default Oracle installations, you can use the following path to find the jdbc driver.

     ```
     C:\Oracle\ora92\jdbc\ojdbc14.jar
     ```
   - **SQLServer**: For SQLServer, use `sqljdbc.jar`. The driver is available from MicroSoft.
2. Copy the jdbc driver for your database type into the `lib` directory under the `librarymanager` directory.

To edit the configuration file(s) for your database

You must edit the configuration file(s) for your database. Some edits are required, and some are optional.

1. Locate the default `config` file for your database in the `librarymanager/conf` folder.

   Note: You can make as many versions of the `config` file as you need. For example, if you have multiple databases for Detect (such as one each for development, quality assurance, and production), you can use a different configuration file for each of them. As a best practice, give the file a name that identifies the database to which it refers. For example, you could name a file that refers to your dev database as `Detect_dev.config`.
2. Edit the configuration file(s) so that the `jdbcDatabaseURL` setting identifies the database to which to connect.

   Note: Although the examples shown here wrap onto multiple lines, in the configuration file each must be on one line.
   - For Oracle, modify it so that it identifies your database server's name (or IP address), its port, and the name of the database.

     ```
     jdbcDatabaseURL=<Detect Database Server name or IP
     Address>:<port>:<database name>
     ```

     By default the port is 1521. If the default port is not used, you can find it in the `tnsnames.ora` file under the Oracle directory in `NETWORK\ADMIN`. It will also give the fully-qualified path required to reach the Oracle server. To test that it is a fully-qualified path, ping it.
   - For SQLServer,

     ```
     jdbcDatabaseURL=//<Detect Database Server name or IP
     Address>:<port>;database=<database name>
     ```
3. If you intend to use the Library Manager to move just a few workspaces, then you can skip this step and instead create empty target workspaces using the Detect user interface before you import the workspaces. However, if you intend to move many workspaces you may find it too labor intensive to create them individually. In that case, use this step to set up the Library Manager to create them for you.

   If you did not install Detect to the default path, or your database is on another machine, modify the following line in this file so that it points to your Detect database directory.

Note: This setting identifies where the database scripts are stored. If they are on a remote machine, you must be able to reference the directory, or the files can be copied locally.

```
db.home.dir=c:/Program Files/Unica/AffiniumDetect/Application/Database
```

Note: You must use forward slashes, as shown in the default path. Also, although this example wraps onto multiple lines, in the configuration file it must be on one line.

4. Determine how you want to manage your database password. You have these options:
   - To allow the Library Manager to connect to the database automatically, modify the following line to replace CEE4 with the correct password.
     ```
     jdbcPassword=CEE4
     ```
   - If, for security reasons, you do not want to store an un-encrypted password in a text file, you can copy the encrypted database password from the registry, or use the Password Encryption tool (`PasswordEncryption.exe`).
   - To have the Library Manager tool prompt you for the password, delete the line or comment it out by prefixing it with the hash mark (#).
5. If you want to change the default directory for log files, modify the following line to specify the correct log file directory. Otherwise, skip this step.
   ```
   logfile.home=C:/TEMP
   ```
   Note: You must use forward slashes, as shown in the default path.
6. You may have more than one configuration file in the `conf` directory so that each configuration file points to a different database. To create multiple configuration files, you can optionally rename `detect.config` when you save changes to this file.

To add arguments to the batch files in /bin (optional)

Optionally edit the batch files in the `/bin` directory under the `librarymanager` directory to include the arguments listed in this table.

`-vendor` *`vendor`*	Specify the 3-character vendor code. Vendor codes are case sensitive. To avoid inconsistency, is recommended that you always use upper case when typing vendor codes. If you do not use this argument, the Library Manager will either present you with a list of existing vendors from which to choose, or if there is only one vendor defined in the database it will automatically select that one.
`-config` *`config.file`*	Specify the name of the configuration file you want to use. The database of the configuration file should point to the database to which to connect. If you do not include this argument, you can open the connection using the Library Manager.

To set logging settings in /conf/log4j.properties (optional)

Optionally edit the `log4j.properties` in the `/conf` directory to adjust the logging level, format, or the location of the log directory.

About using the Library Manager for import and export

You can use the Library Manager to import and export workspaces to and from the Detect database. Workspaces are exported and imported as `.xml` files.

Before you can import or export one or more workspaces, you must open a connection to an Detect database from the Library Manager.

After you make the connection and choose a vendor (unless one is set in the `LibraryManager.bat` file), you can adjust import, export, and audit settings.

To connect to a database from the Library Manager

Use these instructions if you did not modify the `LibraryManager.bat` file with the `-config` option.

1. Double-click LibraryManager.bat file located in the /bin directory under the librarymanager directory. Library Manager appears, along with a command-line window. Do not close either of these windows.
2. Select **File > Open Connection** from the Library Manager to open the **Connection to** dialog.
3. Browse to the conf directory, and select a configuration file you have edited according to the directions for preparing to use the library management tools. The database of this configuration file should point to the database from which you want to export workspaces.
4. Click **Open**.
5. If you left the password setting blank in the configuration file, or if the password has expired, you are prompted for the password. You have the option to save an encrypted version of the password in the configuration file.

To select a vendor in the Library Manager

When you open the Library Manager, it checks to see if there are vendors defined. If there is only one, or if the `-vendor` argument was set in the `LibraryManager.bat` file, then the Library Manager automatically selects that vendor. Otherwise, if there are multiple vendors defined in the database, the tool presents a list of vendor codes from which to choose.

If a list of vendor codes appears, select the vendor code you want and click **OK**.

The Library Manager is populated with a list of the workspaces within the database for the selected vendor. If the workspace's name is different than the display name, the interface shows both names.

To set or adjust Library Manager options

You can set Library Manager options so that it operates with your preference settings so yo do not have to re-enter them every time you run an export or an import. For example, you can set the Library Manager to run the DiffManager every time you do an import. (Most of the options can only be set through the **Options** settings; there is no prompt for them.) After the options are set, the selections are maintained each time you run the Library Manager (if you are using it as the same operating system user).

1. Select **Tools > Options** to open the **Options** dialog box.
2. Set the options on the **Import**, **Export**, and **Audit** tabs, then click **Apply** or **OK**.

The Import options for the Library Manager

The following table describes the Import options for the Library Manager.

Goal	Options	Details
To set the DiffManager to run during every import	Select **Show Diff on Import.**	The DiffManager compares the `.xml` being imported with the database. After the tool runs, you have the option to continue with the import or cancel it. Do not continue with the import if there are changes that you do not want to import.
To set the DiffManager to display all changes, including modification date and annotation changes	Select **Strict Mode**	This setting is equivalent to the `-strict` argument for the DiffManager.
To set which tab of the DiffManager displays by default	Select **Graphical Mode** or **Text Mode as the Default Mode**	The option you select determines which tab is selected by default in the DiffManager's results window. You can switch between tabs to see either display mode. The Graphical Mode shows rules information only, while the Text Mode shows rules and dependencies. **Note:** The format and content of the information in the Text Mode tab is similar to what is captured in the (optional) audit file.
To override the owner of imported rules	Select **Override owner of imported rules with**, then use the drop-down list to select a new owner.	The list shows all of the user names stored in the database.
To preserve the existing names of data sources and fields and not overwrite them during a subsequent import	Select **Do not import names of data sources and their fields; keep what is in the database.**	This setting tells the Library Manager to ignore name changes for data sources and fields during an import. For example, suppose you imported a trigger system from the Trigger Library, but then customized the data source field names. You can use this option if you want to re-import a new version of the trigger system without overwriting the names of data sources or fields.
To verify the schema is defined for all internal lookup tables that are being imported	Select **Verify that all internal lookup tables being imported have their schema defined.**	Lookup tables can require an internal or external connection. If the connection is internal, this option will verify whether the schema for the tables exist in the database. If the verification popup indicates that there is missing schema for a table or tables, you can complete the import but you must define the schema before you can run the engine.

Goal	Options	Details
To set notification when any external lookup table is imported	Select **Notify me about any external lookup table being imported.**	Lookup tables can require an internal or external connection. If the connection is external, this option will notify you that an external table is being imported (whether it exists or not). The tool cannot verify schema for external tables, therefore you must verify that the schema exists.

The Export options for the Library Manager

The following table describes the Export options for the Library Manager.

Goal	Options	Details
To export dependencies for exported workspaces	Select **Export dependencies for exported workspaces**.	This is the recommended setting.
To export only the dependencies that are used by the selected workspace	Select **Limit dependencies exported to only those used by the selected workspaces**.	This is the recommended setting.
To be prompted every time about whether you want to export dependencies for exported workspaces or to limit dependencies to only those used by the selected workspace	Select **Prompt during each export.**	
To set the default export directory	Click **Search** and browse to the directory that you want to use as your default export directory.	This option sets the default export directory, but you can choose another directory after you run the export. This option also sets the default import directory.
To set the encoding for the export file	Select an encoding method from the drop-down list.	This option can only be selected in the Options dialog box; there is no prompt for this option when you run the export.
To export each workspace into its own file (when exporting multiple workspaces at the same time)	Select **When exporting multiple workspaces, put each workspace in its own file.**	The workspace runspace codes are used to name the separate workspace files. This option can only be selected in the Options dialog box; there is no prompt for this option when you run the export.

The Audit options for the Library Manager

The following table describes the Audit options for the Library Manager.

Goal	Options	Details
To store audit history when you import workspaces	Select **Store Audit trail of imports in**, then click **Search** to browse to the directory in which to store the audit trail for imported workspaces.	The audit file provides a Unix-style diff for changes to all objects being imported.

To use the Library Manager for export

Follow this procedure to use the Library Manager to export a workspace from your database to an `.xml` file on your computer.

Note: Functionality of the Export, Import, and Audit options for the Library Manager are controlled using the **Tools > Options** settings.

1. Open a connection to a database from the Library Manager.
2. If necessary, select a vendor.
3. Select one or more workspaces listed in the Library Manager to export.
   - You can filter the list of workspaces by typing the name of one or more workspaces in the **Filter By** field. Separate workspace names by a comma or a semicolon (;), but no spaces. Workspace names are case sensitive. You can use an asterisk (*) as a wildcard character.
   - To select a single workspace to export, click the workspace folder.
   - To select a second workspace (when the first workspace is already selected), press and hold the **CTRL** key then click anywhere within the white space on the row that contains the workspace.

   Note: You cannot click on the folder icon or the name (when the CTRL key is pressed). Instead, click to the right of the workspace name, or in another column. To select more than one workspace in a row, hold the shift key down and select a range of workspaces from the list.
4. Select **Tools > Export** from the Library Manager.

   The **Export to** dialog appears.
5. Save the exported `.xml` file.
6. If the Library Manager is set to prompt you when exporting dependencies, you are prompted for your preferences.
7. When a dialog indicates the export is complete, click **OK**.
8. Select **File > Exit** from the Library Manager to close and exit the Library Manager.

To use the Library Manager for import

Follow this procedure to use the Library Manager to import a workspace from an `.xml` file to the database.

Important: If you are importing many new workspaces, then ensure that you edited the configuration file(s) to point to the directory for your Detect database.

Note: Before you import a file that contains workspaces that are already in your database, you should compare the workspace in the file to the workspaces in your database. You can set the Library Manager to run the DiffManager automatically. Alternatively, you can run the DiffManager manually to compare import files with the database prior to importing them. You.

Note: Functionality of the Export, Import, and Audit options for the Library Manager are controlled using the **Tools > Options** settings.

1. If you are importing many new workspaces, then ensure that you edited the configuration file(s) to point to the directory for your Detect database. Otherwise, if you are importing a few new workspaces, you can use the Detect user interface to create empty workspaces with the same codes in the target database.
2. Connect to a database from the Library Manager.
3. If necessary, select a vendor.
4. Select **Tools > Import** from the Library Manager to open the **Import from** dialog.
5. Navigate to the exported files.
6. Select the file containing the trigger information you intend to import, and click **Open**.
7. If the DiffManager is not set to run automatically, skip to Step 9. Otherwise, when the Diff Manager appears, examine the information it presents.
8. If there are changes that you do not want to import, close the DiffManager and click **No** to cancel the import. Otherwise, click **Yes** to continue with the import.

   Depending on the size of your imported data, you may experience a waiting period before the import is completed. Status messages appear at the bottom of the window.
9. When the **Import complete** dialog appears, click **OK**.
10. Select **File > Exit** from the Library Manager to close and exit the Library Manager.

About using the command-line to import and export workspaces

You can use a command-line interface for exporting database workspaces into an `.xml` file, and for importing workspace information from an `.xml` file into an Detect database.

To use the command-line interface for exporting workspaces

To use the command line to export workspaces from a database, run the `ExportManager.bat` file located in the bin directory under the `librarymanager` directory. Use the following arguments to specify the database connection and workspace data.

Argument	Description
`-vendor` *`vendor`*	Specify the 3-character vendor code. (Required) Vendor codes are case sensitive. To avoid inconsistency, is recommended that you always use upper case when entering vendor codes.

Argument	Description
`-config` *config.file*	Specify the name of the configuration file that you want to use. The database setting of this configuration file must point to the database from which you want to export workspaces. (Required)
`-file` *file.xml*	Specify the name of the xml file to which to export the workspaces. (Required)
`[-workspaces` *workspace1|workspace2|...*`]`	Specify the 3-character runspace codes for the workspaces to export in a pipe-separated list. You can use % to match all. (Optional, unless you do not specify workspace dependencies for export) Note that runspace codes are case sensitive, so T01 is not the same as t01.
[-dependencies]	Optionally specify this argument if you want to export workspace dependencies. (Optional, unless you do not specify a workspace for export)
[-limitDependencies]	Optionally specify this argument if you want to limit the exported workspace dependencies to only those used by the exported workspaces. (Optional)
[-help]	Optionally specify this argument by itself to view the help for this command. (Optional)

For example, you can type the following line to export vendor `ABC` workspaces `A01` and `B05` to the `exp.xml` file, using the database configuration specified in `export.config`.

```
ExportManager.bat -vendor ABC -config ../conf/export.config -workspaces
A01|B05 -file exp.xml
```

To use the command-line interface for importing workspaces

To use the command line to import workspaces to a database, you must run the `ImportManager.bat` file located in the `bin` directory under the `librarymanager` directory. Use the following arguments to specify the database connection and workspace data.

Argument	Description
`-vendor` *vendor*	Specify the 3-character vendor code. (Required) Vendor codes are case sensitive. To avoid inconsistency, is recommended that you always use upper case when entering vendor codes.
`-config` *config.file*	Specify the name of the configuration file that you want to use. The database setting of this configuration file must point to the database from which you want to export workspaces. (Required)
`-file` *file.xml*	Specify the name of the xml file from which to import the workspaces. (Required)
`[-user` *user_name*`]`	Optionally specify this argument to assign ownership of the imported rules and components to a user. (Optional)
`[-help]`	Optionally specify this argument by itself to view the help for this command. (Optional)

For example, you can type the following line to import workspaces in the `exp.xml` file located in a directory named exported, using the database configuration specified in import.config for vendor `ABB`.

```
ImportManager.bat -vendor ABB -config ../conf/import.config
                  -file ../exported/exp.xml
```

Important: If you are importing many new workspaces, then ensure that you edited the configuration file(s) to point to the directory for your Detect database.

Note: Before you import a file that contains workspaces that are already in your database, you may want to compare the workspace in the file to the workspaces in your database.

About comparing import files with the database prior to import

You can use Detect tools to compare workspaces in an .xml file to those of your database before you import workspace data into the database.

You can use the command-line interface, or you can use the Detect Diff Manager to see the differences between workspaces you intend to import and those in your database. See the following topics:

- "To use the command-line for import file comparisons"
- "To use the Diff Manager for import file comparison"

After you compare a workspace file to your database, if you are satisfied that importing this file is beneficial to your database, you can import the file using the Library Manger or from the command line.

To use the command line for import file comparisons

To use the command line to compare workspaces in a file to those in a database, you must run the `DiffManager.bat` file located in the `bin` directory under the `librarymanager` directory. Use the following arguments to specify the database connection and workspace data.

Argument	**Description**
`-vendor` *`vendor`*	Specify the 3-character vendor code. (Required). Vendor codes are case sensitive. To avoid inconsistency, is recommended that you always use upper case when entering vendor codes.
`-config` *`config.file`*	Specify the name of the configuration file that you want to use. The database setting of this configuration file must point to the database from which you want to export workspaces.
`-file` *`file.xml`*	Specify the name of the xml file containing the workspace information for comparison. (Required)
`[-outputdir` *`output_dir`*`]`	Optionally use this argument to specify the directory in which to place the workspace comparison results. (Required if you do not specify the `-g` argument)
`[-strict]`	Optionally specify this argument to view all changes, including modification date and annotation changes. If you do not use this argument, modification date or annotation changes are not displayed. (Optional)
`[-g]`	Optionally specify this argument to use a graphical user interface (Diff Manager) to view workspace differences in a graphical tool before you import. (Required if you do not specify the `-outputdir` argument)

Argument	Description
[-help]	Optionally specify this argument by itself to view the help for this command. (Optional)

For example, you can type the following line to compare the workspaces in the `exp.xml` file located in a directory named `exported`, to those in the database configuration specified in `import.config` for vendor ABB. In this example, you assign the difference information to be stored in the directory named `CompareToData`, located under the `librarymanager` directory.

```
DiffManager.bat -vendor ABB -config ../conf/import.config -file
../exported/exp.xml -outputdir ../CompareToData
```

To save workspace differences in an output directory

If you use the `-outputdir` argument in `DiffManager.bat`, all workspace differences are saved in a subdirectory of the directory whose name you specify for the `-outputdir` argument. The subdirectory name is assigned the root name of the file you specify for the `-file` argument. In the previous example, the comparison data is stored in the `../CompareToData/exp` directory. If you run the comparison more than once with the same arguments, a number is appended to the subdirectory name. The comparison information appears under the subdirectory as follows:

- Changed rules appear in a directory called `diffs`. Each changed rule appears in its own text file.
- Rules that are in a workspace in the import file that are not in the workspace in the database are listed in a file called `newRules.txt`.
- Rules that are in a workspace in the Detect database that are not in a workspace of the same runspace code in the import file are listed in a file called `missingRules.txt`.

To use the Diff Manager for import file comparison

The Diff Manager is a graphical tool you can use to compare the workspaces you intend to import into a database with those that already reside in the database. To use the Diff Manager, you must use the `-g` argument with `DiffManager.bat`.

For example, to compare workspaces in the `exp.xml` file located in a directory named `exported`, to those in the database configuration specified in`import.config` for vendor `ABB`, you would use the following syntax:

```
DiffManager.bat -vendor ABB -config ../conf/import.config -file
../exported/exp.xml -g
```

You can compare the workspaces for import to the workspaces in the Detect database using the following Diff Manager features:

- Changed rules appear under the **Changed Rules** tab in the left pane of the Diff Manager.
- Rules that appear only in a workspace in the workspace import file and not in the workspace in the database, are listed under the New Rules tab.
- Rules that are in a workspace in the Detect database that are not in a workspace of the same runspace code in the import file are listed under the **Missing Rules** tab in the left pane of the Diff Manager.

- If you select a changed rule on the left pane of Diff Manager, the changes to the rule are displayed in the right pane of the Diff Manager according to the rule structure as follows/
  - Properties–Any rule property changes appear in yellow under the Properties tab, with the Properties tab appearing in yellow in the right pane.
  - Events–Any rule event changes appear in yellow under the Events tab, with the Events tab appearing in yellow in the right pane.
  - If–Any changes to if statements in the rule appear in yellow under the If tab, with the If tab appearing in yellow in the right pane.
  - Then–Any changes to then statements in the rule appear in yellow under the Then tab, with the Then tab appearing in yellow in the right pane.
  - XML Source–The XML source code for the rule appears under the XML Source tab. The starting point of changes to the selected rule source appear in red under the XML Source tab.

Chapter 10. Preparing Data Files

IBM Unica Detect uses three types of data files.
- Transaction Data
- Profile Data
- Database Lookup Tables

Detect processes data based on an entity association, a common key that associates related data sources. Entity examples are a Customer ID, an Account ID, a Household ID.

The following conventions apply to entity associations.
- Within the Detect application, an entity is named with a single lower-case character.
- A single entity can have multiple related data sources.
- Data sources that are related are defined with a common identifier field.

About transaction files

Transaction files contain the data that was captured by an operational system. Transaction files are in ASCII flat file format.

File format for transaction files

The format of a transaction file must meet the following guidelines.
- One header record followed by zero or more records.
- All record lines have delimiters between fields, but not before the first field in the line or after the last field in the line.
- The delimiter is a single character that is configurable in the Configuration Utility. The pipe character (|) is recommended because it appears less frequently in real data than other popular delimiters.
- The delimiter must not be used within data fields.
- A pair of delimiters next to one another indicates the absence of data for the field positioned between those two delimiters. If the first field of the record contains no data, then the line begins with the first delimiter, and if the last field contains no data, then the line ends with the last delimiter.

Field names for transaction files

The field names on the header line must be defined in the Data Source Manager, for the designated data source. The spelling and case of the field names must match exactly. One field must be nominated as the common identifier field in the data source. This field is common on all the data sources that are associated to the given entity. The ID field must be in a packed format. It is suggested that the ID field have a fixed length, and that the actual values be padded to the maximum length.

Data line records in transaction files

The data line records must be alpha-sorted on the field nominated as the identifier (the entity ID). Within transaction files, there may be multiple data line records for each unique entity ID. Those records must be sorted by transaction date and time within each entity ID.

About naming transaction files

Transaction files and flat file profile files use the following naming convention:

```
unica.<entity type>.<file name>.<YYYYMMDDhhmmss>
```

Note the following.

- The prefix used can be any of these values: `unica`, `detect`, `insights`, `marketsoft`, or `elity`. The example above uses Unica as the prefix.
- For transaction feed files, the date on the file extension should be the date of the earliest transaction in the file. However a few transactions in the file before this date are acceptable. For example, for ramp up, files containing data from a month or week are often used. These files should have a date from the first of the month, even if they contain a small percent of "catch up" transactions from the previous month.
- The entity type is always a single letter.

Restriction: NULL values in fields in transaction files

The Detect is unable to match fields with NULL values when performing comparisons. You must substitute either a single printable character or a character string for the null values if you want to be able to recognize that condition in comparisons.

Restriction: Date fields in transaction files

Dates fields must be in system-readable date format even if you want those dates to be empty. In other words, dates cannot be NULL. Null dates must be represented as either very old dated or very future dates.

Here is an example of a transaction file:

```
ID      |NAME    |CALLED_NUMBER|CALL_LENGTH|TRAN_DATE_TIME
001234|David     |732-123-4567       |15                  |2003-02-10 09:12:33
001234|David     |732-111-5555       |48                  |2003-02-10 10:11:50
002941|Jeremiah|732-777-8888       |40                  |2003-02-10 11:22:44
005555|Anthony |732-333-4444       |27                  |2003-02-10 03:01:02
005555|Anthony |732-32-8945         |121                |2003-02-10 10:12:30
005555|Anthony |973-597-0022       |2                    |2003-02-10 19:00:21
006789|Tom       |732-111-2222       |4                    |2003-02-10 06:54:01
```

About profile files

A profile file contains information that describes the designated entity rather than what the entity does. Examples of the type of information contained in a profile file are age, home address, and telephone number.

Profile data is not a requirement for an entity. However, if profile data is not defined, qualifier components cannot be created for the entity. When a profile is not described, the descriptive information needs to be included in each transaction record if it is needed as a criteria of a trigger system.

Profile data can be read from database tables, from an ASCII flat file format, or from a combination of database tables and a flat file feed.

Note: The flat file format provides the highest engine throughput.

An entity associated with profile database tables must have a connection string to the table; it must also have one field designated as the key. This field is the common identifier on all data sources associated to the given entity.

An entity defined as having a flat file profile must have that file included with each engine run. A flat file profile must have the same characteristics as the flat file described above for transaction files.

There must be a single data line record for each unique ID in the transaction files. Transaction file records without an associated profile record are ignored. Profile records without an associated transaction file record are ignored. In both instances, extraneous records impact processing efficiency.

Here is an example of a simple profile file:

```
ID|AGE|ZIP
001234|25|11111
002941|55|22222
005555|31|33333
006789|60|44444
100382|18|55555
```

About naming profile files

Transaction files and flat file profile files use the following naming convention:

unica.*<entity type>*.*<file name>*.*<YYYYMMDDhhmmss>*

Note the following.

- The file name is the Source Name in the Data Source Editor.
- The prefix used in the source name can be any of these values: `unica`, `detect`, `insights`, `marketsoft`, or `elity`. The example above uses `unica` as the prefix.
- The entity type is always a single letter.

Note:

Event Enabled Time Queue (EETQ) Profile Files

An Event Enabled Time Queue (EETQ) profile file contains profile data for inactivity processing. A time queue profile is required when both of the following are true:

- an entity is defined with a flat file profile file
- the workspaces contain Forward Inactivity components

The Event Enable Time Queue manages inactivity. Its purpose is to wake-up at the end of an inactivity period (Forward Looking Inactivity) and generate an event for each user reaching this point.

In order to process the inactivity events an EETQ profile feed is required for each entity type that is associated with a Profile Feed File data source. This additional EETQ profile feed is used by the engine in conjunction with all the inactivity events it recognizes to be acted upon for the particular run.

The EETQ profile feed is an additional file. It is in addition to the profile feed, database tables, and transaction feed files associated with the entity type. It contains profile records for each user that is significant to the inactivity events for a given run.

If the profile file has rows for every possible entity, then the EETQ profile file is not required. It is needed if a profile is defined and you are only feeding records for the entities that transact that run. It contains profile information for non-transacting entities that have an inactivity come due that run.

The EETQ profile feed must use the same layout as the profile feed file for the associated entity type. It DOES NOT contain a header line. Each system run requires only one EETQ profile feed per entity type regardless of the dates on the individual transaction feed in the batch.

To create EETQ profile files

1. Use the Events Preprocessor tool to create a list of entity IDs.
2. Use the list created in step 1 to create the EETQ profile feeds.
   - Each entity type needs its own EETQ profile feed
   - The EETQ profile feeds must use the same layout as the entity profile feeds
   - The EETQ profile feed does not contain a header line.
3. The EETQ profile feeds must be named:

   `unica.EETQ.<entity type>`

   Note the following.
   - The prefix used in the source name can be any of these values: `unica`, `detect`, `insights`, `marketsoft`, or `elity`. The example above uses `unica` as the prefix.
   - The entity type is always a single letter.
4. Put the EETQ profile feeds in your designated feeds folder.

   Note: You do not need a time queue profile file when profile data for users for whom inactivity might be coming due is included in the main profile file.

About date and timestamp on profile and transaction feed files

The profile feed file and transaction files always contain a date stamp as part of the name. (A time stamp should also be included in the name of the transaction feed file, and optionally on the profile feed file.) This date/timestamp is a reference point for engine processing indicating the order in which the files are processed. Files with the same date/timestamp are fed simultaneously. A profile, if employed, needs to be created for each set of files with a different date/timestamp.

If the system administrator does not define a data source to have a field whose `field app type` is `transaction date`, then:

- Feeder uses the date/timestamp in the file name to group and process all files with the same date/timestamp value
- date/timestamp value in the file name is used as the starting point for engine processing
- Engine assumes that all transactions are in exactly the correct order
- Engine processing uses the date/timestamp value in the file name when evaluating the significance of the transaction and for storing history

If the system administrator does define the data sources with a field whose `field app type` is `transaction date`:

- Engine processing recognizes the actual date in the transaction when evaluating the significance of the transaction.

About database lookup tables

Data destined for processing can also reside in Lookup tables. Lookup tables are not entity related.

A Lookup table:

- Can reside in either an external database or in the Detect database.
- Must have one field designated as a key.
- Contains data that is considered non-transactional.

Some possible uses for a lookup include a list of all typecodes that represent a deposit, or a list of all retail item SKUs associated with products that are on sale during a particular week. Lookup tables can also be used to define profile information.

About connection strings

Every database table, regardless of whether it is a Profile table, a Lookup table, internal to the Detect database or in an external database, requires a connection string. Connections that are internal to Detect are created using the Configuration Utility during the installation process. External connections that are needed for vendor-specific tables can be created using the **Data Sources > Connections** interface.

Chapter 11. Using the IBM Unica Detect Tools

This section provides a brief description of the following IBM Unica Detect tools:

- **Feed Validator—**Scans feed folders and individual files and produces a report about the feed status.
- **Engine Trace Utility—**Provides a means of debugging trigger systems by interpreting the trace log output by the Engine.
- **Ramp-up Engine—**Provides a means of pre-loading containers with data for specific users.
- **Event Preprocessor—**Generates a file of entity IDs that are associated with inactivity events.
- **Entity State Optimizer —** Cleans out stale data from the state tables for more efficient running of the Detect engine.
- **Outcome Management Tool—**Provides a way to reformat outcome records for Detect triggers so that the fields in the outcome records can be accessed and used by other products.
- **Password Encryption Tool—**Provides a way to encrypt passwords to be used in the database connections table. You can also use the tool update the registry with an encrypted password.

Note: There are additional tools described elsewhere, including the Library Management tool and the Configuration Utility.

About batch files

Some Detect components and utilities can run one of several different batch files when they complete a run. There are different batch files for each component or utility and for each process result: success, success with errors (for some processes), and failed.

Default, empty files are shipped with the product; you must edit these files as needed to perform the appropriate actions for your system when a Detect process finishes. If the batch file for a process result (as specified in the Configuration Utility) is empty, no further processing takes place.

The processes that can run a batch file when they finish are:

- Events Preprocessor
- Rampup Engine
- Outcome Management Tool (OMT)
- Detect engine (batch and always-on runs)

When you run from the command line, the `/ftp` switch determines whether the the process invokes the batch file when it finishes. To have the process run the batch file, use `/ftp:On` in the command you use to run the process. When you run the Detect engine from the Engine Manager, check the **Batch Notification** checkbox to run a batch file when the process completes.

Batch file usage examples

For example, after an engine process runs successfully, the success file could call the OMT. When the OMT process finishes successfully, the OMT success file could call the product that will use the outcome of the OMT.

Another example would be for when a process fails to complete. In that case, you could have its failure batch file send a notification of the failure to the appropriate people.

Batch file names

In the Configuration Utility's Advanced Configuration settings, in the Notifications section, you can see the default names of the batch files. By default, the file names specified in the Configuration Utility are the names of the files provided with the Detect installation. If you change these names in the Configuration Utility, you must also change the names of the files themselves or create new files with these names.

Batch file required location

The batch files that Detect processes invoke at the end of runs must be located on the machine where the Enterprise Management Controller (EMC)is installed.

About the alert.bat file

In addition to attempting to run files associated with success, success with errors, and failure, the Detect engine can run an alert file N seconds before the end of the run. This applies only to always-on runs of the Detect engine,

This feature is configurable. This file is executed if you set the if the `Alert Before Processing Completion` option in the Configuration Utility's **Advanced Configuration > System Options** section to a value greater than `0`. The value you set specifies the number of seconds before processing completes when Detect attempts to run the file.

Process results that run batch files

The following is a list of processes and process states that cause Detect to attempt to execute a batch file specific to a process outcome.

Engine run, batch mode
- Transaction success
- Transaction success with errors
- Transaction failure

Engine run, always-on mode
- Always-on initialization success
- Always-on initialization failure
- Always-on shutdown success
- Always-on shutdown failure
- Always-on transaction success
- Always-on transaction success with errors
- Always-on transaction failure
- Always-on, all transaction runs (`alert.bat` file)

Events Preprocessor run
- Preprocessor success
- Preprocessor failure

Rampup Engine run
- Rampup success
- Rampup failure

Outcome Management Tool (OMT) run
- OMT success
- OMT failure

About the Feed Validator tool

The Feed Validator tool is a stand-alone executable that can be used to verify the integrity of transaction feeds that the customer provides to Detect. The Feed Validator scans feed folders and individual files and produces a report about the feed status. The report contains information about the following types of problems:
- missing feed files
- misspelled field names
- missing or out-of-sequence fields
- incorrectly formatted data in the fields

Any of these errors can cause Detect to fail to run.

The typical time to use this tool is during the early implementation phase, when you are introducing a new feed file type (data source).

The tool can be used by one user at a time, but many instances of this tool can work simultaneously.

Note: This tool can only run on a machine that has Detect installed. Also, the tool needs the database instance containing the Detect metadata to compare to the feed files.

To run the Feed Validator

The Feed Validator executable is `FeedValidator.exe`). It is located in the `Application\bin directory` under your Detect installation.

You may want to create a shortcut for this program the first time you use it.
1. To run the tool, enter `FeedValidator.exe` at the command prompt.

   The **Feed Validator** tab is for checking that the necessary files are in the folder, as well as the header on the individual file(s).
2. Enter the path for the file or folder for the tool to validate.
3. Continue to the subsequent sections for the steps for the following tasks:
   - "To validate header and file content" on page 72

- "To validate for correct sorting"

To validate header and file content

The header is invalid if it is missing a field that is defined in the data source fields in the database. However, the header may contain more fields than defined in the data source fields in the database.

The following steps describe how to validate the header and optionally how to validate the file content as well.

1. On the **Feed Validator** tab, enter the **Rule User** name.

   The rule user name is the database user name where the rule schema exists. It is the prefix before a table for SQL command such as in the following example, which has RuleUser as the rule user name:

   select * from RuleUser.fid__dsmdatasources
2. Enter your Vendor ID in the **Vendor** field.
3. Enter your Password and Login information.

   You can use this tool if you are a valid user for the vendor. The password and login are set within the Detect database.
4. Enter the delimiter used in the file(s). This is the specific delimiter the feed files use to separate fields
5. If you want to validate the file content, check the **Check Content c**heck box. Checking this box sets the tool to look at both the header and the actual transaction/profile information. If the checkbox is cleared, the tool validates only the header. If checked, the tool attempts to verify that the data in the fields are of the same data type defined in the data source. For example, that the fields contain strings or numbers as required.
6. To verify that data source defined in the system are named in the folder, click **Validate Feed Folder**. Note the messages that appear in the Output area.
7. To check that the spelling and case of field names in the feed files match those that are defined in the system, click **Validate Feed Files**. Note the messages that appear in the Output area.

To validate for correct sorting

1. Choose the **Check Sort** tab.

   The **Check Sort** tab is for checking that the file is sorted properly.
2. Enter the **Entity ID Field** name. This is the name of the field that the file should be sorted by.
3. In the **Unsorted Limit** field, specify how many errors to report before the tool stops validating.(Zero means scan all the way through the file.)
4. Click **Check for Sort** to start the process. You can stop the process at any time by clicking **Stop Checking**.

The results are displayed and are logged to a file.

About the Engine Trace Utility

After you build the Detect trigger systems, and set up the data sources, you may have to debug the trigger systems and data sources to run Detect correctly.

You can use the Engine Trace Utility to examine and interpret the trace log of Detect events into a format that is useful to the business or services person in charge of making the application work:

- The information appears in a hierarchical (tree) format, starting with an entity at the top level.
- The hierarchy of information displayed includes the transactions processed by the entity, and the events that follow.
- You can expand and collapse the hierarchy to facilitate navigation.
- Trouble areas are highlighted to indicate possible points of failure.

In order to successfully use this tool, you must understand the following:
- Detect components
- trigger systems
- transactions being processed

When to use the Engine Trace Utility

You should use the Engine Trace Utility if, when you run Detect after defining the components and trigger systems and entering the data sources, nothing happens, or the system produces the wrong results .

You can use the Engine Trace Utility to uncover mistakes in the logic of the trigger system.

Before you do so, you can use the Feed Validator Tool to verify the formatting of the transaction feed files.

Before running the Engine Trace Utility

Before you run the Engine Trace Utility, there are a number of topics that you must understand.

Trace log complexity settings

The trace log generated by the system can have varying levels of complexity, as determined by the debug mode setting. Each message is assigned a level and is only inserted into the log if the engine's current debug mode is equal to or higher than the messages debug level.

You can control the debug mode two ways:
- When running the engine from the Engine Manager's user interface.
- When running the engine from the command line.

Note: Generating the trace log represents a significant performance load, but you should use this feature only in test environments or when troubleshooting a production workspace when performance is not an issue.

Performance considerations

Opening large trace logs could take considerable time.

Error handling

An error message is displayed if you choose a file other than a trace log produced by the engine or one that contains unrecognized entry values or tags.

Engine Trace Log grammar

The Engine Trace Log has its own grammar, described in Appendix C, "Grammar Basics of the Engine Trace Log," on page 131.

To run the Engine Trace Utility

1. Run the Engine Trace Utility from the following executable:

   `enginetraceutility.exe`

   You can find this file in the `Application` > `bin` directory of your Detect product installation directory. The Engine Trace Utility appears.

   Note: This executable requires that the files `trace.dll` and `traceschema.txt`. be in directory with it. If you have moved the executable to a non-default directory, be sure to also copy `trace.dll` and `traceschema.txt` into the same directory.
2. Select and open one or more trace logs.
3. Select an entity from list of entities found in the logs.
4. A tree view appears that has the RUN node as its root.
5. If necessary, expand the tree view may to expose details of each transaction processed for this entity.
6. To get more information, select nodes of the tree view to reveal extra details of the node on the right side of the screen.
7. Examine the list of trouble spots, and use that area of the screen to navigate to each trouble spot.
8. Once you determine the cause of the problems, fix them, and rerun the engine.

To open Trace files

1. Click **Open** on the toolbar of the Engine Trace Utility to open a **Browse for Folder** dialogue box.
2. Select a folder log and click **Ok**. The Select Trace Logs dialog appears.
3. Select one or more files, move them to the **Open Logs** list, and click **Ok**.

To select an entity

After opening the trace log, the **Select an Entity** dialog box appears.

Entities are the subject of the transaction being processed, such as the user, account, or household.

1. Type an Entity ID to reveal a list of sorted IDs that begins with the typed entry.
2. Select an entity and click **OK**.
3. The tree view fills with nodes from the entity's trace log entries. The nodes are the heart of the utility. Each node represents a single entry in the trace log.
4. To change entities click the **Entities** button to reopen the dialog box.

To reveal run details

Selecting the RUN node reveals information about the run in the Details box.

Note: Refer to the comments within the traceSchema.txt file for formatting options for the Details box.

To reveal profile information

Selecting the ENTITY node reveals profile information in the details box.

You can expand and collapse the tree to reveal more nodes. Select an entity to expand a list of transactions for it.

To reveal transaction information

Select a TX (transaction) node to reveal the transaction values in the details box.

The values displayed for the transaction (and profile) in the details box only include fields that are actually in use by the engine. Fields that have not been referenced in any components are not sent to the engine so they will not show in the details box.

To understand trouble spots using the Engine Trace Utility

Select a trouble spot in the list to reveal and highlight the trouble spot on the tree view. Information about the Trouble Spot appears in the details box.

Interpreting the trouble spots requires knowledge of the triggers and transactions defined for your system. Through the trouble spots list, the Trace Utility can draw attention to places where experience has shown that triggers can unexpectedly halt progress and generate outcome different from what is expected. The trouble spots list does not represent where things went wrong, only where things may not have gone as expected.

A short list of examples:
- Trigger controls prevent a trigger from firing more than a defined number of times. The limit might be enforced earlier than expected, or the existence of the trigger limit may have been forgotten.
- Rule durations (start and end dates, period modes) might expire, preventing a trigger from firing.
- Some triggers can have a reset event that restarts the state of the trigger for that entity. The firing of a reset event can cause a trigger to not fire when expected.

About the Ramp-up Engine

The Ramp-up Engine provides a means of pre-loading specific containers with data for specific users. The goal of this process is to save time in building up state history for the client portfolio base so that the system can produce outcome sooner rather than waiting for the history to accumulate over time.

If a user already has state history for the container, then the Ramp-up Engine either replaces or appends to the existing data. If the user does not have state history for the container, then a new entry is created.

The Ramp-up Engine is typically run once.

Before running the Ramp-up Engine

Before running the Ramp-up Engine, you must create, name, and put the container feed files where they belong.

Also, if the system initialized for always-on operation, you must issue the `shutdown` command as described in “Running a production workspace from the command line” on page 100.

About naming the container feed files

Prior to running the Ramp-up Engine you must create a feed file for each specific container that is to be ramped. Each file should have the name:

`unica.<entity type>.<container ID>.<timestamp>`

Note that the entity type is always a single letter.

Set the feed file timestamp to the earliest date in the feed, just as for engine feeds.

About dividing large container feed files

If your configuration uses multiple cluster units and you have a very large feed file (80 million accounts, for example), you can break up the feed file so that there is one portion for each of the cluster units. If the cluster unit has multiple engines running, then the same hash algorithm that is set in the Configuration Utility will distribute the load to the various engines within the cluster unit.

About formatting the container feed files

The container files have a header line and data line. The header line in the feed file must contain the fields that match the field names of the Type Descriptor used in the container. The field delimiter must be the one defined in the Configuration Utility's Advanced Configuration settings.

The first field of every record in the file should be the entity ID. In the header line, use the name `entityID` for the first field.

The records in the file must be sorted in ascending order, by entity ID. If there are multiple records for the same entity ID, sort the multiple records by date-timestamp.

About placing the container feed files

Each cluster unit has a feed folder assigned to it using the Configuration Utility. Put all of the container feed files for a cluster unit in that feed folder. The folder can contain feed files for any entity type.

About running the Ramp-up Engine

The Ramp-up Engine can be run from the command line or invoked from a batch file.

Running the Ramp-up Engine from the command line:
To run the Ramp-up Engine from the command line, execute the following statement with the appropriate commands:

```
rampup.exe
```

The following tables describe the required and optional commands.

Table 9. Required commands

Command	Description
`/ven:`*`vendor`*	A 3-character vendor code
`/ven:`*`vendor`*	A 3-character vendor code
`/workspace:`*`workspace_name`*	The workspace `pub` is the default. **Note:** Workspace names are case-sensitive.
`/login:`*`user`*	Detect administrative user
`/password:`*`password`*	The password of the Detect administrative user

Table 9. Required commands (continued)

Command	**Description**
`/behavior:`*`append`* or *`purge`* or *`insert`*	For this parameter, *`append`* and *`purge`* indicate what to do with the container contents if one exists for the entity ID. **WARNING**: do not use *`insert`* if state information already exists; instead use either *`append`* or *`purge`*. Use the *`insert`* option only when you are absolutely sure the state table is empty. The *`insert`* option is an optimization for the fist time that state is loaded. It is very efficient for first time state ramp up.

Table 10. Optional commands

Command	**Description**
`/ftp:ON` (or `OFF`)	Batch notification setting. The default is `ON`. When it is set to ON, this parameter tells Detect to execute the files associated with the notification settings in the Configuration Utility's Advanced Configuration settings. (Log into the Configuration Utility as an administration user.) • Transaction Failed Action • Transaction Success Action • Transaction Success with Errors Action Depending upon the outcome of the run, Detect will execute one of the files associated with the parameter signaling an end of run. If the parameter is set to `OFF`, then the process will not execute.
`/recover`	Used to recover from a previous run

How the Ramp-up Engine processes

When the Ramp-up Engine executable runs, it calls the EMC which calls the TSM and then loads the cluster definitions and starts the ramp-up engine for each engine on each cluster unit.

The Ramp-up Engine processes all files in the specified ramp up directory. All specified containers are loaded during initialization. The Ramp-up Engine then processes among all files, in ascending order, by entity ID, loading state history only once for each entity ID. Records for entity IDs that are not in ascending order are bypassed during processing, and are written to a log file.

The `behavior` parameter determines whether a container is cleared before new data is saved in the container, if the new data is appended to the existing data, or if new data is inserted. If no container exists in state history for the entity ID, the container is created and the new data is saved. The volume of data that is saved in each container during the ramp up adheres to the Overflow characteristic associated with that container.

About bookmarking

Similar to other Detect tools, the Ramp-up Engine creates a bookmark to indicate that a entity ID has ramped successfully. Should a run fail, this bookmark can be used in the recover mode to indicate that the tool should re-start from this point.

About logging

Each Ramp-up Engine will log to a folder under your Detect installation, as follows:

`<vendor>\logs\ RampUpEngine_<Cluster ID>_<Instance ID>`

About the Events Preprocessor Tool

The purpose of the Events Preprocessor is to generate a file of entity IDs that are associated with inactivity events for the next EETQ processing run. The output file creates a list of IDs that you can use to create the EETQ profile feed. Output is generated in the form of a file containing a list of significant entity IDs.

You can enter input options using the events preprocessor interface, or from the command line.

To run the Events Preprocessor tool

By default, the Events Preprocessor tool is located in the `Application\bin` directory under your Detect installation. It is recommended that you create a shortcut for this tool the first time you use it.

1. Run the following file to display the Events Preprocessor screen:
   `EventsPreProcessor.exe`
2. Enter the following input options:
   - **Vendor**
   - **Role**
   - **User Name**
   - **Password**
   - **Entity Type**

   Note: For multiple entity types, enter a comma-separated string or run the executable multiple times.
3. Enter the path for the Output Directory. The path specifies where the following files are written:
   - output file—Contains a list of significant entity IDs. The output file is named `unicaEETQuserIDs.`*`entity type`*
   - log file—Contains a count of the number of significant entity IDs
   - error log file—Describes errors that occurred while running the tool.
4. Enter the **Start Date** and **End Date** in the format YYYY-MM-DD hh:mm:ss or YYYY/MM/DD hh:mm:ss.

   Note: The hh:mm:ss portion of the date parameters is optional, and will default to 00:00:00 if not specified. The system will look for data one second before the date specified.

   The tool will select entity IDs that are due for a wake up call beginning with and including the Start Date, up to but not including the End Date. For example, a start Date of *2002-03-30* and an End Date of *2002-03-31* will select entity IDs for the 30th.
5. Click **Start** to run the tool. Previously created output files are overwritten.

An "All Good" message in the Result box indicates that the preprocessor has completed successfully. An error message in the Result box indicates that the preprocessor did not complete successfully. Check the error log for details when this occurs.

6. Click **Close**.

To run the Events Preprocessor from the command line

You can run the Events Preprocessor from the command line, or you can invoke it from a batch file.

To run the Events Preprocessor from the command line, execute the following statement:

```
eventspreprocessor.exe
```

You can include any of the event preprocessor parameters when you execute the Events Preprocessor.

Events Preprocessor parameters

Command	Description
`/start`	Runs the Events Preprocessor from the command line.
`/v:`*vendor*	Specifies the 3 character vendor code.
`/r:`*workspace*	Specifies the workspace. The workspace `pub` is the default.
`/U:`*user*	Specifies the system admin user employed by Detect to access its system tables.
`/P:`*password*	Specifies the password of the system admin user.
`/type:`*entity type*	Specifies a single character entity code. You can enter multiple entity codes, separated by commas.
`/dir:`*folder*	Specifies the location of the output files.
`/Sdate:` *YYYY-MM-DD*	Specifies the start date in `YYYY-MM-DD` format.
`/Edate:` *YYYY-MM-DD*	Specifies the end date in `YYYY-MM-DD` format. If you want data for all of 12/8, use 12/9 for an end date.

About cleaning the state tables using the Entity State Optimizer

You can use the Detect Entity State Optimizer to clear stale or unimportant data from your Detect state tables for a given workspace. You can run this tool with its user interface, or from the command line.

Running the Entity State Optimizer with the user interface

The Entity State Optimizer has a user interface that allows you to select processing options and run the tool. The following procedures describe its use.

- “To run the Entity State Optimizer with its user interface” on page 80
- “To run the Entity State Optimizer in audit-only mode” on page 80
- “To run the Entity State Optimizer to clean state tables” on page 81

- “To optionally clean batches of the state table in parallel” on page 81
- “To recover processed records if the tool stops before completion” on page 82

To start the Entity State Optimizer

1. Navigate to the `Application > bin` directory of your Detect installation directory.
2. Double-click `Entity State Optimizer.exe`.
3. Log in to the database with the appropriate Detect administrative user name and password. The Detect Entity State Optimizer appears.

To run the Entity State Optimizer with its user interface

1. Start the Entity State Optimizer.
2. Select the workspace or runset from the **Workspace/Runset** list.
3. Check each of the **Clean Options** you want to use:
   - **Clean Containers —** To clean state data associated with container rules
   - **Clean patterns —** To clean state data associated with pattern rules
   - **Clean Forward Looking Inactivities —** To clean state data associated with forward-looking inactivity rules
   - **Remove all records with empty states —** When you choose this option, the following items are removed.
     – Any records in the state table whose state data would be emptied as a result of selecting any of the other three **Clean Options**.
     – Any records in the state table whose state data is already empty.
4. Select a process date from the Calendar that appears when you select the **Process Date** field. The process date that you use is the date used to calculate the end date for the event horizon (state age) for your Detect engine rules. For each rule of the type(s) you select for cleaning under **Clean Options**, (pattern rules, container rules, forward-looking in activity rules), the associated state data is identified as stale (slated for removal) if it is older than the event horizon calculated for the rule.
5. Run the Entity State Optimizer for these settings in audit-only mode.
6. Run the Entity State Optimizer for these settings to clean the state tables.

To run the Entity State Optimizer in audit-only mode

1. Start the Entity State Optimizer.
2. Select the workspace or runset from the **Workspace/Runset** list.
3. Check each of the **Clean Options** you want to use:
   - **Clean Containers —** To clean state data associated with container rules
   - **Clean patterns —** To clean state data associated with pattern rules
   - **Clean Forward Looking Inactivities —** To clean state data associated with forward-looking inactivity rules
   - **Remove all records with empty states —** When you choose this option, the following items are removed.
     – Any records in the state table whose state data would be emptied as a result of selecting any of the other three **Clean Options**.
     – Any records in the state table whose state data is already empty.
4. Select **Audit Only** mode. This setting allows you to test the state table data without affecting it.
5. Select **State Table** (default) for the source of the entity IDs.
6. Click **Start**.

When the audit completes, you can see the results displayed under **Audit Statistics**. In addition, you can find a file listing the entity IDs to be updated in the `UpdateEntities.log` file, which is located under the `vendor_name > logs > ESO` directory in your product installation directory. Note that this directory path is not created until your run the Entity State Optimizer audit for the first time.

You can use the Entity State Optimizer audit log files as a basis for selecting entity IDs from which to clean the state tables when you select **File** for the **Source for Entity IDs**.

To run the Entity State Optimizer to clean state tables

Follow this procedure to clean the state tables after you run the audit.

1. Start the Entity State Optimizer, maintaining the settings you used for runnign it in audit mode: **Clean Options**, **Workspace/Runset**, and **Process Date**.
2. Select **File** for the source of the entity IDs, and click the **Browse** button located next to the **File Name** field, to navigate to and then select the UpdateEntities.log file, which is located under the *`vendor_name`* `> logs > ESO` directory in your product installation directory.

   Note: When you use the audit log file records for entity ID selection, you restrict the search of the state tables for cleaning purposes to only those records listed in the selected file. These are exactly the records that satisfy the cleaning criterion specified in the audit. Consequently, using the audit file for the entity ID selection renders the state table record search more efficient when you run the Entity State Optimizer to clean it.
3. Select the **Clean Only** mode. When you do so, the Entity State Optimizer does not re-run the audit, and this makes the processing more efficient.

   Note: You may also select **Audit and Clean** mode to run the audit, display and log the audit results, and to clean the state tables. When you do so, the Entity State Optimizer overwrites the audit file located under the *vendor_name* > logs > ESO directory in your product installation directory. Consequently, if you select **Audit and Clean**, back up the audit files under a different name (or in a separate directory) as an added safety precaution.
4. Click **Start**.

   Note: If you have a large number of records listed in the audit log file, you can optionally open multiple instances of the Entity State Optimizer and simultaneously run them on smaller subsets of the records listed in the audit log files. In this case, you should select a run-time folder path.

To optionally clean batches of the state table in parallel

After you create an audit log file in the Entity State Optimizer, you can examine the audit log file. If your audit log file contains a large number of records, you can run the Entity State Optimizer on subsets of the record data in an audit log file. You can also run multiple instances of the Entity State Optimizer simultaneously, running the batches in parallel.

To clean a batch of records listed in an audit log file, follow this procedure for each instance of the Entity State Optimizer you open:

1. Start the Entity State Optimizer.
2. Use the same **Workspace/Runset**, **Clean Options**, and **Process Date** settings that you use to create your audit log file. These settings are retained by default, each time you open a new instance of the Entity State Optimizer.

3. Create a unique run folder (`Run_Folder`) for the batch, under the `vendor_name` > `logs` > `ESO` directory in your product installation directory.

   Note: If you want to create and retain the audit data for each batch of records you intend to clean, you must create a different run folder for each batch. Otherwise, the audit data is overwritten in the run folder you specify in the Entity State Optimizer. If you do not specify a run folder in the Entity State Optimizer, and you run an audit, the audit data in the `vendor_name` > `logs` > `ESO` directory is overwritten.
4. Create two text files as follows:
   a. Open a text file editor and create `batchstart.txt`. In this file, enter the starting record number. For example, if you want to start the batch with record 1000, enter 1000, and save the text file in the `Run_Folder` directory you created in Step 3.
   b. Open a text file editor and create batchend.txt. In this file, enter the ending record number. For example, if you want to end the batch with record 1500, enter 1500, and save the text file in the `Run_Folder` directory you created in Step 3.
5. Select the run folder you create for this batch in Step 3, by clicking the **Browse** button located next to the **Folder Name** field.
6. Optionally select **Audit and Clean** mode to retain per-batch audit logs. Otherwise, select **Clean Only** mode.
7. Click **Start**.

To recover processed records if the tool stops before completion

If the Entity State Optimizer fails to complete auditing and/or cleaning the state table, or if you stop it before it completes the process, you can find the entity ID of the last record that is successfully processed by the Entity State Optimizer.

To do this, follow these steps:
1. Navigate to the `vendor_name` > `logs` > `ESO` directory (or to the *vendor_name* > logs > ESO > *run_folder* directory, if you created one), and locate the `bookmark.txt` file. This text file contains the record of the last record successfully processed by the Entity State Optimizer. This file is only present if the Entity State Optimizer audit and/or clean process stops before completion.
2. After examining this record, you can optionally clean batches of the state table in parallel to complete the Entity State Optimizer audit and the clean process on the remaining records.

Running the Entity State Optimizer from the command line

You can run the Entity State Optimizer from the command line to perform the same tasks you can perform from the ESO user interface.

Example command

The following example command uses only the required parameters, not any of the optional ones. Note the quotes around the name of the tool. These are required due to the spaces in the tool name.

```
"Entity State Optimizer.exe" –start –u:someUser –p:somePassword –w:ABC
```

Command line reference

The following table details the parameters you can use to run the ESO from the command line, without invoking the user interface. For full descriptions of how the ESO works, see the procedures detailing its use from the user interface.

The commands are shown with a forward slash (/) to start each command. However, note that you can also use a hyphen (-) to start commands.

Command	Description
`/start:` *`processing_mode`*	Runs the Entity State Optimizer from the command line; the ESO user interface does not appear. Also sets the processing mode. If you do not specify a mode, the mode defaults to `AUDIT`. The options are: • `AUDIT` • `AUDIT_COMMIT` • `COMMIT` Setting this parameter is the equivalent of selecting an item in the **Mode** group box in the user interface. The following options are required with `/start`. • `/u` • `/p` • `/w`
`/u:`*`user_name`*	Required when running the ESO without the user interface. Specifies a user for authentication purposes. The user must be a Detect system administrator.
`/p:`*`password`*	Required when running the ESO without the user interface. Specifies a password for authentication purposes. Must be the password of the Detect system administrator.
`/w:`*`workspace`*	Required when running the ESO without the user interface. The 3-character workspace or runset code. Setting this parameter is the equivalent of selecting an item from the **Workspace/Runset** drop-down list in the user interface.
`/date:`*`process_date`*	Optional. Sets the timestamp for aging entity states. The value of *`process_date`* must be a date string using the ISO-8601 date format (`YYYY-MM-DD`). If *`process_date`* is omitted, the current system date is used. Setting this parameter is the equivalent of setting the **Process Date** control in the user interface.
`/clean:`*`clean_option`*	Optional. Defines which resources are cleaned. The options are: • `all`—All resources are cleaned • `b`—Backward Looking Inactivities are cleaned • `c`—Containers are cleaned • `f`—Forward Looking Inactivities are cleaned • `p`—Patterns are cleaned You can enter multiple options, using no spaces or other separator.
`/delete`	Optional. Causes any states that are completely cleared of all data to be deleted. If you do not use this flag, empty states are not deleted. Setting this parameter is the equivalent of checking the **Remove All Records with Empty Stores** checkbox in the user interface.

Command	Description
`/filename:`*`file_name`*	Optional. Specifies a pre-existing file containing a list of entity IDs to process. If you do not use this flag, the ESO uses the `STATE` table as the source of the
`/runfolder:`*`file_path`*	Optional. Specifies the folder where the ESO creates logs and working files. If you do not use this flag, the ESO uses the *`vendor`*`\logs\ESO` folder. If the folder does not exist, ESO creates it.Setting this parameter is the equivalent of setting the **Folder Name** text box in the **Run-Time Folder for Batch Optimization** group box in the user interface.

About managing outcome for use by other products

The Outcome Management Tool (OMT) utility allows you to process outcome records for Detect action triggers so that the fields in the outcome records can be accessed and used by other products. With this utility, the processing of outcome information takes a fraction of the time necessary for the Detect engine to generate it.

Who uses the OMT

Note: Only Detect users with administrator rights can log into the OMT.

Users of this feature might be a:
- Detect user who designs triggers
- Detect user who defines trigger aliases and mapping names within the OMT. This Detect user should coordinate with the user of the product that will be using the outcome to make it clear which aliases and mapping names should be used.
- Detect user who will run the tool to process and produce the trigger tables
- Detect administrator to maintain the trigger tables and aliases

About the Outcome Management Tool (OMT)

The OMT manages the structure of the outcome record for a particular trigger. Specifically, the OMT parses the multiple values embedded in the current XML message field in the outcome and uses them to populate trigger tables so that each value has its own field in a database table.

The OMT also provides the option to create a more condensed packaging of the tabular outcome data (such as information stored in containers) so that the data is stored as text in one delimited field in the main outcome table, rather than as separate fields in a separate table.

OMT gives you the ability to provide a new name (alias) for each piece of data included in the message field so that the field names are recognizable to downstream processes or to people viewing them.

After the OMT outcome tables are created and tested, the OMT tool is typically run in batch mode or from the command line. There is no associated scheduler feature. The OMT should be run after Detect runs and generates records. When the OMT is run, it creates new records by parsing the existing records. Users can subsequently access the records in either format.

Note: Alias definitions will be imported and exported when the Library Management Tool is used.

The OMT:
- Maps Action outcome values to trigger table names
- Creates trigger tables
- Populates trigger tables with values parsed from the outcome message field
- Truncates trigger tables
- Audits trigger tables, comparing them to Action components

Note that by default the OMT processes the set of rows associated with a single run ID in the outcome table only once.

About installation and configuration

The OMT is automatically installed when you install Detect.

Where to find the OMT

The Outcome Management Tool (OMT) is named `OutcomeManagerTool.exe`. It is located in the `Application\bin` directory under your Detect installation.

You may want to create a shortcut for this program the first time you use it.

Database requirements

The user of the outcome schema must have bulkadmin permission to be able to populate the OMT tables.

To set bulk admin permissions for SQL Server

1. In SQL Server Management studio, go to **Security > Logins**.
2. Right click on the outcome schema user and select **Properties**.
3. On the Server Roles page, ensure the **bulkadmin** checkbox is checked.

   Note: For Oracle, you do not need to adjust any settings as long as you followed the installation instructions on for setting directory security in the *Detect Installation Guide*.

About OMT Workflow, with use case

This section describes the general workflow for using the Outcome Manager Tool (OMT). The example uses Campaign as the tool that will access the fields in the outcome records.

1. Detect designer creates triggers and runs Detect to generate trigger outcome information.
2. The OMT user then selects the workspace for which to process the outcome records.
3. The OMT user assigns aliases to the main trigger tables, container or select tables, and fields. The aliases are intended to provide meaningful and recognizable field names for the person setting up the product (such as Campaign) that will use the data.
4. The OMT user generates the empty tables using the OMT.

5. The OMT user runs an audit, which creates a text file that describes the status of the table to trigger relationships. If the audit confirms that the tables are ready to process, the next step is to populate the tables.
6. The user of the other product sets up their product to access the tables. For example, the Campaign user might map to the trigger outcome tables.
7. Detect user runs Detect to generate trigger outcome data.
8. The OMT tables are populated so that the outcome data is written to the database trigger tables. The OMT tables can be populated in one of two ways.
   - During testing phases, the OMT user will populate the tables using the OMT.
   - During production the OMT will typically be run from a batch file that is set to start automatically after Detect runs successfully. (By default that file is ftpput.bat and it is identified in the Configuration Utility in the Advanced Configuration tab.)
9. The trigger tables are available for use by another tool. For example, a Campaign flowchart designer maps these Detect trigger tables and can select them as input to a flowchart process (for example, in a Select process).
10. Detect or Campaign user initiates processing of Detect outcome into Campaign trigger tables. This process can be automated.
11. When you are done, clear the outcome tables.

About workflow when changes are made to trigger outcome

Note: When changes to the trigger system affect the outcome of the trigger, you must ensure that the changes are managed within the OMT and are communicated to those who are responsible for managing the products using the outcome. There is no automatic notification or alert. Most likely you will have to remove from the outcome table all records for that trigger that were generated before the change.

This section provides an example workflow for when the trigger system has been modified.

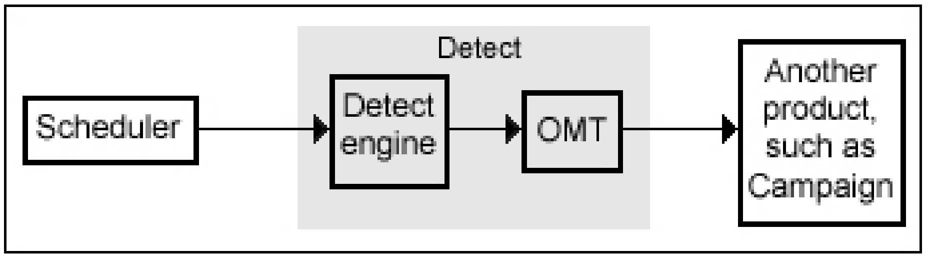
1. Detect designer modifies one or more of the triggers in a workspace.
2. The OMT user logs into the OMT and if the OMT tables have already been created, the user drops the tables.
3. The OMT user updates any aliases or changes settings as needed, then saves the changes.
4. The OMT user runs an audit, makes adjustments as needed in the OMT, then regenerates the trigger tables for the modified trigger(s).
5. The OMT user communicates the changes to the designer for the tool(s) that are accessing the outcome tables.
6. A designer for the tool using the outcome makes adjustments within that tool so that it can process the new trigger tables.

About workflow when the trigger is deleted from Detect

Before you delete a trigger which has OMT tables associated with it, use the OMT to drop the associated tables. Otherwise, when you delete the trigger the tables will be orphaned.

How to incorporate OMT into automated processing

After the aliases have been defined and the OMT outcome tables have been created and tested, the OMT can be integrated into an automated process to populate the tables as shown in the following diagram.



1. Before processing, batch files must be set up for Detect success and failure as well as OMT success and failure. For example, after Detect runs successfully, the `ftpPut.bat` file should call the OMT and the `omtDone.bat` file should call the product that will use the outcome of the OMT.
2. Set up a scheduler to run edmdriver.exe to run the Detect engine.
3. The scheduler starts the Detect engine.
4. When Detect completes successfully, it calls the Transaction Success Action file (`ftpPut.bat`), which calls the OMT.
5. When OMT completes successfully, it calls the OMT Success Action batch file (`omtDone.bat`).

About the tables generated by the OMT

This section provides details about the tables generated by the OMT.

Outcome types

Outcome values in Detect are of two types:

- **scalar outcome values** – for example, timestamp, customer ID, rule ID of the Action, etc. These values are passed out of Detect as a single value string and are stored in the main trigger table. One field is created in the main trigger table for each scalar value.
- **tabular outcome values** – additional information from containers. These values can be stored in additional trigger tables associated with the main trigger table. One trigger table is created for each tabular value in the outcome. In other words, one main trigger table is created for each component, and multiple additional tables can be associated with that main trigger table.

Location of the tables

Tables will be generated in the location defined by the Detect outcome schema. The OMT tables reside with Detect outcome tables schema.

About the main trigger tables

Each trigger in an Detect workspace will populate a main trigger table with information from Detect's outcome table. This table will be named based on the alias, which may be the default alias created by the OMT or one defined manually within the OMT. The main trigger table will contain the following fields:

- `OUTCOMEID` – A unique ID for each outcome. It is the primary key on the trigger table
- `RUNID` – The run ID
- `ENTITYID` – The ID of the entity that generated the outcome

- `TIMESTAMP` – The date and time identifying when the outcome was generated
- `TEXT` – The text field that appears in every outcome trigger. This text is the free-form text from the Action editor
- Other fields – Additional fields may be included, based on the definition of the trigger. Each field is named using the alias designated for it in the OMT. If the **As Scalar** setting was used, then there is an additional text field that contains the comma separated list. The name of that field is created from the Tabular alias.

About additional trigger tables

Tabular information, such that coming from a Container or Select component, is provided in a table separate from the main trigger table. Each separate piece of tabular information that the trigger generates will have its own table.

As a best practice, create an alias that makes it obvious that the additional table is related to the main table. For example, *`<trigger alias>_<tabular alias>`*.

Every additional table contains the following fields:

- `OUTCOMEID` – A unique ID for each outcome. This field is a foreign key to the same field in the main trigger table.
- Additional fields – Additional fields are included based on the definition of the tabular component using the proper datatype(s). Each of these fields will be created using the alias defined in the OMT.

About the Audit file

The audit function provides valuable information about the status of the relationship of the components to the tables that were created from the outcome. For example, it might identify that tables have not been created or that there are inconsistent table definitions.

The audit compares the trigger table alias information in the Action components to existing trigger tables. The audit reports one of the following states for each trigger:

- **Ready to Process**: the trigger table exists and matches the information in the Action component
- **Inconsistent Table Definitions**: The trigger table exists, but does not match the alias information in the Action component
- **Tables Not Created**: The alias information has been defined, but the trigger table has not been created
- **Incomplete Trigger Definitions**: The Action component is missing some or all alias information

The results of the audit can be saved in a report for dissemination of any changes to Campaign designers. It is recommended that this report always be created prior to generating the trigger tables.

About the user interface

The user interface for the OMT provides key information about the status of the aliases and tables for the selected workspace.

Note: Functionality is limited when the PUB or XPB workspaces are selected. Only the Populate Tables, Clear Tables, and Audit buttons are available and nothing can be changed in the definitions.

About the trigger list

You can expand the trigger list to show the outcome values associated with that trigger.

About the color codes

Color-coded circles and text mark each element in the selected workspace to indicate the status of each.

- Red: definition is incomplete
- Yellow: inconsistent information. For example, alias definitions and table definitions do not match.
- Yellow with an X: table is missing
- Green: ready to process
- Red text: the alias is not saved

About the tool buttons

The tool buttons act upon the current workspace.

Note: Functionality is limited when the PUB or XPB workspaces are selected. When either of those workspaces is selected, only the **Populate Tables**, **Clear Tables**, and **Audit** buttons are available and nothing can be changed in the definitions.

- **Save –** Save the changes, such as changes to the alias names.
- **Generate Tables –** Create OMT tables using the definitions.
- **Populate Tables –** Populate the OMT tables with data. This button is used during manual testing. In production, OMT tables are populated in batch processing.
- **Clear Tables** – Removes data from the OMT tables.
- **Drop Tables –** Drop the OMT tables. Use this button to drop the tables created by the OMT. If the tables have already been created for a workspace, you should drop the tables before you make any changes. For example, if you change aliases for a trigger, the OMT will create new tables for that trigger and orphan the old ones. To avoid having orphaned tables, drop the tables first, then make and save your changes.
- **Audit –** This button generates an audit file.

About the Workspace menu

The **Workspace** menu has options that match the functionality of the tool buttons and act upon the selected workspace. The **Workspace** menu has these additional options:

- **Accept Defaults –** returns all alias settings to their default values. (To save the aliases, use **Save Changes**.)
- **Refresh** – refreshes what you see inside OMT data. The response is the same as if you changed to a different workspace and back again. If a new trigger was added in Detect while the OMT is still open, you must log out of OMT and log back in to refresh the trigger list.
- **Save Changes** – saves your changes.
- **Discard Changes –** discards your unsaved changes.

About the Trigger menu

The **Trigger** menu has options that match the functionality of the tool buttons and act upon the selected trigger.

About the Alias list

The OMT creates default aliases, using the name of the triggers, containers, and fields. You can keep the default aliases or you can optionally change the alias to something more recognizable to the people who will be setting up the product that will use the outcome. This list describes where the aliases are used:

- **Trigger Alias –** this alias is used as the name of the main trigger table created by the OMT.
- **Tabular Alias –** this alias may be used for one of two things. If the **As Scalar** setting is selected, the alias is used as the name of the text field in the main trigger table that contains the comma-separated list of values from the container or select table. If the As Scalar setting is not selected, the alias is used as the name of the table created to hold the container or select data.
- **Field Alias –** this alias is used as the field name in a container table.
- **Scalar Alias –** this alias is sued as the field name in the main trigger table.

About the As Scalar setting

Only **Tabular** selections can be marked **As Scalar**. If the **As Scalar** setting is selected, the OMT combines the fields from the container or select table into a comma-separated list of values within a field in the main table.

About the Alias Status

The Alias Status column indicates whether any of the aliases are not valid and will describe the problem with the alias.

About the Table Status

The Table Status column indicates whether the table is ready to be processed, missing, or inconsistent.

About running the OMT from the user interface

Typically the OMT is run from the user interface for two reasons:

- Setting up the aliases and generating the tables
- Testing

To run the OMT from the user interface

1. Start the OMT.

   The name of the the Outcome Management Tool (OMT) executable is `OutcomeManagerTool.exe`. By default, it is located in the `Application\bin` directory under your Detect installation.
2. Log in as a valid Detect user.
3. To create your own aliases so that they provide meaningful and recognizable names for the person setting up the product that will use the data, perform the steps desribed in the procedure "To map aliases."
4. Click **Generate Tables** to generate the empty tables.
5. Click the **Audit** button to create an audit file which describes the status of the table to trigger relationships. If the audit confirms that the tables are ready to process, go to the next step.
6. Use the **Populate Tables** button for test purposes. In production, the OMT outcome tables are populated in batch processing.

To map aliases

Use this section if you want to create your own aliases so that they provide meaningful and recognizable names for the person setting up the product (such asCampaign) that will use the data.

Note: If the table has already been generated, drop the table before changing an alias. Otherwise, you will have orphaned tables.

1. From the drop-down list, select a workspace for which to process the outcome records.

   The action triggers in the workspace appear in the tree, and details about each appear to the right.
2. Examine the list of aliases.
   - Change any alias that you think should be more recognizable
   - Change any alias that is marked with an Alias Status other than Valid.
3. Click **Save**.

To set up another product to use the tables

The user of the other product sets up their product to access the tables. For example, the Campaign user would map to the trigger outcome tables. The exact process for setting up another tool to process the outcome tables depends on the product that will use them and beyond the scope of this document.

Best practices suggest that you provide them with the audit file.

To populate the tables

Use the **Populate Tables** button for test purposes. In production, the OMT outcome tables are populated in batch processing.

About running the OMT from the command line

You can run the Outcome Management Tool (OMT) from the command line, or you can invoke it from a batch file. In either case, you must either run it on the machine where the EMC is, or you must identify the machine using the /server parameter. To run the OMT from the command line, execute the following statement:

```
OutcomeManagerTool.exe
```

You can include any of the parameters shown in the table below when you execute the OMT.

OMT Parameters	**Description**
/run	Tells the OMT to just start processing outcome instead of opening the user interface
/ftp:ON (or OFF)	Batch notification setting. When it is set to ON, this parameter tells Detect to execute the files associated with the notification settings in the Configuration Utility's Advanced Configuration settings. (Log into the Configuration Utility as an admin user.) • OMT Failed Action • OMT Success Action Depending upon the outcome of the run, Detect will execute one of the files associated with the parameter signaling an end of run. If the parameter is set to OFF, then the process will not execute.
/server:*<server>*	The machine where the EMC can be found
/user:*<username>*	The Detect user
/password:*<password>*	The Detect user's password

OMT Parameters	Description
`/vendor:<vendor>`	The vendor code. Vendor codes are case sensitive. To avoid inconsistency, is recommended that you always use upper case when entering vendor codes.
`/workspace:<workspace>`	The code for the workspace to be processed

After the OMT finishes processing

When the command line utility completes processing outcome, it executes one of two scripts.

- `OutcomeDone.bat` if processing completed successfully
- `OutcomeError.bat` if the utility encountered errors while processing

The files are included with the installation of Detect, but the contents must be set by the customer and depend on what actions they wish to be performed upon success or failure of outcome processing.

OMT folder under the Detect installation directory

Populating the OMT creates files in the `Application\OMT` directory under your Detect installation.

Scalability of the OMT

For scalability, the OMT divides trigger IDs based on the number of outcome records each one has. Multiple OMT instances operate on separate blocks of trigger IDs distributed as evenly as possible across the engines.

When the OMT executable runs, it calls the EMC which calls the TSM and then starts the OMT engine for each engine on each cluster unit. (The cluster definitions are set in the Configuration Utility.)

Recoverability of the OMT

OMT processes a list of trigger IDs. It checks the Detect trigger tables and joins the OMT outcome trigger tables and other OMT outcome tables based on outcome ID. If it has already processed an outcome ID then it skips that trigger when processing. If you experience a failure, contact IBM Unica Technical Support.

About the Password Encryption tool

Note: The Password Encryption tool is installed on the IIS machine only.

By default, the Password Encryption tool (`PasswordTool.exe`) is located in the `Application\bin` directory under your Detect installation.

Use this tool when a database password has changed, and you need to update the encrypted password in the registry.

To update the password in the registry, enter the password into the **Password to be Encrypted** field, then click **Encrypt**. To update the registry with the encrypted password, click **Update Registry**.

The encrypted password is the same one that is generated in the connections of the Configuration Utility.

Chapter 12. Running Workspaces and Runsets

IBM Unica Detect supports two run modes: batch (the only mode until the release of version 8.2.0) and always-on. See “About always-on mode” for information about this mode.

The Detect engine can call a configurable batch file at the end of a run. See “About batch files” on page 69 for more information.

Engine Manager or command line?

During development and testing of trigger systems, you can run a Detect workspace in batch mode using the Engine Manager pages in the Detect web application, or from the command line.

You must use the command line to run the production workspace (pub), and to initialize, run, or shut down any workspace you run in always-on mode.

You can stop any run, including always-on runs, either through either the Engine Manager or the command line.

The Engine Manager is typically used for testing workspaces and runsets under development. Before you promote a workspace or runset to production, it is common practice to validate the accuracy of new components and trigger systems by creating specific use cases and feeding transactions crafted to fire those cases.

About connection pools

The system’s default connection pool size is 5. This default value should be sufficient for loading workspaces and runsets.

However, if you see an error that says "Timeout expired," the timeout period elapsed before the system obtained a connection from the pool. This may have occurred because all pooled connections were in use and max pool size was reached. If this happens, you may increase the value for `MaxPoolSize` in the registry on every engine server.

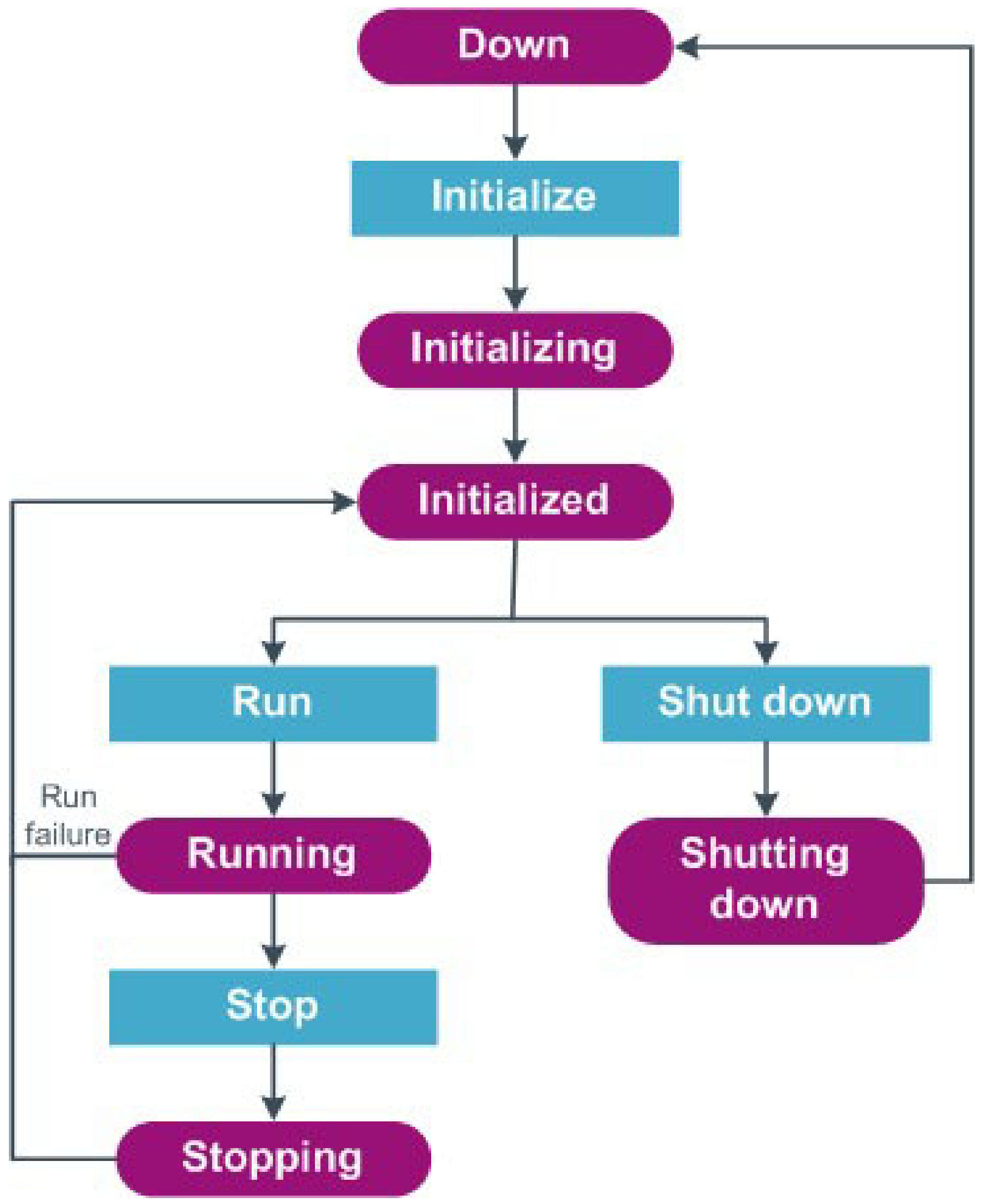
Do not increase the pool size unless you experience the error. Increasing the pool size can substantially increase the total number of connections used, which can affect performance.

About always-on mode

In always-on mode, you can execute multiple runs without going through the start-up and shut-down process. Running in always-on mode allows Detect to run continuously and process events rapidly, in nearly real-time fashion.

To run the engine in always-on mode, you must run from the command line.

The following diagram illustrates the always-on states and process flow.



Always-on mode operates as follows.

- You issue commands through the command line to initialize, start, stop, and shut down runs. You can also use the Engine Manager to stop an always-on run, but not to issue the other commands.
- An always-on run that is stopped remains initialized, and you can start another run immediately.
- To exit always-on mode, you must issue the shut down command.
- After you have initialized a workspace run, you must shut down the run if you want to run a different workspace. You can run only one workspace at a time in always-on mode, unless you have configured multiple environments.
- When multiple environments are configured, each environment can run in either always-on or batch mode. Each of these environments can run different workspaces.

If you plan to operate Detect in always-on mode, you should size your environment to handle peak or near-peak transaction volumes.

Log file location

When Detect runs in batch mode, the log files are all created in their own directories, as described in "About Detect log files" on page 143. When you run in always-on mode, an `AlwaysOn` sub-directory is created under these standard directories, and always-on logs are located in these `AlwaysOn` directories.

Monitoring the always-on environment

You can obtain information about an always-on run in two ways: on the Engine Manager page, and by examining the Detect database tables.

On the Engine Manager page:

- If you have started the initialization process, you can see the workspace name in the drop-down list or workspaces. When initialization is complete, the workspace name has **Up** next to its name, and the **Get Run** button is disabled.
- If you have started the run, the **Get Run** button is enabled and you can obtain the status.

In the Detect database, you can examine the `State` column of the `G_ENVIRONMENTS` table in the Detect database. The numbers in this column represent the run state, as shown in the following table.

State	Value
Running	6
Run ended successfully	7
Run failed	8
Run is stopped	9
Run is shutting down	10
Run is stopping	12
An attempt to stop a run failed	13
An attempt to shut down a run failed	14
Initialization failed	15
Run does not exist	18

Setting alerts for always-on run completion

You can configure Detect to run a batch file before the end of a run. For example, you can use this feature to kick off an ETL process at some point during each always-on run, to start creating the feed files for the next always-on run.

To configure this feature, you use the following configuration properties in the **System Options** section on the Advanced tab in the Configuration Utility.

- `Estimate Processing Time`—Turns on the alert feature
- `Alert Before Processing Completion`—Specifies how long before the end of the always-on run that the batch file is run
- `Progress Recalculation Interval`—Specifies how often the calculation is performed

For details on what each option does, see the descriptions that are displayed when you select the option, or refer to Advanced Configuration Settings in the Configuration Utility.

When you have configured these three properties, Detect runs the batch file specified by the `Alert File Name` option, located in the **Notifications** section on the Advanced tab of the Configuration Utility. See About batch files for more information.

For example, you would use the following settings to start an ETL process 4 – 5 minutes before the end of each always-on run, with a calculation interval of one minute.

- Write the necessary commands in the batch file specified in the `Alert File Name` option.
- Set `Estimate Processing Time` to `YES`.
- Set `Alert Before Processing Completion` to `300`, to run the batch file when the system estimates that 5 minutes remain before the run completes.
- Set `Progress Recalculation Interval` to `60` to recalculate every minute.

When to shut down an always-on run

You may want to shut down an always-on run for the following reasons.

- A failure occurs from which the system cannot recover, such as a power outage or loss of database connection.
- The environment changes, such as the adding another server, performing server maintenance, changing the registry, or changing a database password used by Detect.
- You want to update the current workspace. For example, you must shut down and re-start if you want to turn on the recoverability option after you have started running in always-on mode.
- You want to run a different workspace.

In addition, if you change one of the following configuration values in the Configuration Utility, you must shut down and re-start an always-on run.

- `ageContainer`
- `cacheMode`
- `cacheSize`
- `databaseName`
- `delimiter`
- `entityPartitioning`
- `errorLogChannel`
- `feederErrorTolerance`
- `initTimeLimit`
- `LCID`
- `logLogChannel`
- `outcomeDatabase`
- `outcomeSchemaName`
- `performanceLogChannel`
- `portNumber`
- `ruleFiringCount`
- `shutdownTimeLimit`
- `TSMTimeToLive`
- `vhSize`
- `bookmarkBatchCount`
- `floatSignificantDigits`
- `packetSize`
- `feederValidateData`
- `feederThrottle`
- `maxMessagesToSend`
- `feederSleepRoom`
- `feederSleepNoRoom`
- `eetqQueryTimeout`
- `excessiveTransactionThreshold`
- `eetqFeedPath`
- `floatFieldPrecision`
- `integerFieldPrecision`

About state recoverability in Detect

In recoverable mode, Detect processes the feeds from the state at which the system previously halted, and completes the run from that point. This avoids the firing of duplicate outcomes, or corrupting states in the state table. In addition, the ID code for the recovered run remains the same as that for the previous (failed) run.

If you run with recoverability, you must leave the feeds in your feed folder when you run the engine.

You should run with recoverability only if the system fails to finish a previous run successfully.

If you run reports that aggregate data based on the ID of a run, these reports include data for the original run and its recovered run.

You can set the Detect recoverability feature in one of the following ways:
- In batch mode only, you can check the **Recoverability** option when you run the engine from the Engine Manager.
- In either batch mode or always-on mode, you can use the `recover` flag when you run the engine from the command line.

About inactivity mode settings

If you use inactivity events in your Detect engine rules, then you can set the inactivity mode in the Configuration Utility's Advanced Configuration settings. (Log into the Configuration Utility as an admin user.)

The inactivity mode setting determines how Detect responds to inactivity processing.

About cleaning up after the run

You must clean up after each successful Detect engine run.

Depending on your requirements you can archive or delete the following items:
- Files in your designated feeds folder
- Rows in your outcome table

Running or testing a workspace from the Engine Manager

You can use the Engine Manager to obtain the following information about a Detect workspace run:
- The workspace being tested
- The transaction feeds and profile data
- The level of involvement of state history
- Only for runs that use transaction feeds, an estimate of what percentage of the run is complete and the time remaining until completion

To run or test a workspace from the Engine Manager

This procedure applies to runs using batch mode only.
1. Select **Engine Manager** from the navigation bar.

   The Engine manager page appears with the Start New Run tab selected.

2. Select a workspace from the drop-down list.
3. If you have set up multiple environments and associated them with enabled clusters, select an environment from the **Environments** drop-down list.

   This field does not appear if you have not set up multiple environments.
4. If your configuration uses clusters, then you do not need to set the Feed Directory or Entity Data types fields because they are set by default. If you want to override the cluster settings, make sure the entity list is correct and that the feed directory has a valid path.
5. If your configuration does not use clusters, then enter a **Feed Directory** path and select one or more **Entity Data** types.

   Hold down **Control** to select multiple items.
6. Select the **Inactivity End Date**.

   The date that you specify is the date that the system uses to look for any forward looking inactivity events that need to be triggered.
7. Select the desired **Run Parameters**.

   For example, you can choose to clear the knowledge of prior to running the test.
   - **Clear State History –** Erases all retained state information (for example, pattern state and state history).
   - **Recoverability –** The run can recover its state if a previous run failed or was stopped for some other reason.
   - **Batch Notification –** Detect executes the files associated with the following Notification Path Links settings on the Advanced Configuration page in the Configuration Utility.
     - System Success
     - System Error
     - System Success With Errors

     Depending upon the outcome of the run, Detect executes one of the files associated with the parameter signaling an end of run. If you set the option to Off then the process does not execute.
   - **End Event –** Use the drop-down list to select the end event. You can choose the following.
     - **Off –** Select this option if you do not want an end event.
     - **End of Run Event –** If you choose this option, an artificial transaction (ATX) is created at the end of each transaction set for each ID being processed. You can specify an artificial transaction as a significant event in a Pattern component or in a Container Manipulator component.
     - **End of Day Event –** Chooe this option only if you are running the Engine Manager for ramp up purposes. If you choose this option, an artificial transaction (ATX) is created at the end of each day's set of transactions for each ID being processed. You can specify an artificial transaction as a significant event in a Pattern component or in a Container Manipulator component.
8. Set the **Trace Log Mode**. The choices are Off, Trace, and Debug.
   - **Off –** No transaction processing is done by the back engine.
   - **Trace –** Trace the transactions through the back engine and trigger systems. This is the recommended setting.
   - **Debug –** Write additional engine-specific debug information to the log. This setting typically used only at the request of IBM Unica Services.

9. To override the inactivity mode set in advanced configuration, select the checkbox and chose from the following options in the drop-down list.
   - Off
   - Disabled
   - CRM
   - RealTime 1
   - RealTime 2

   See the topic "Task: Adjust system preferences using Advanced Configuration settings" in the "Configuring Multiple Cluster Units" chapter of the *IBM Unica Detect Installation Guide* for descriptions of these options.
10. Click **Start Run**.

    If you want to stop the run when it is in progress, click **Stop Run**.

    Once the run has started, the Run Status section appears at the bottom of the screen. The Run Status section informs you of the progress of a run. As a run progresses, a Log Entry is displayed for each step of the process. A successful run displays these Log Entries:
    - The system is not initialized
    - The system is initializing
    - The system is initialized and is ready to start processing
    - The system is processing
    - The system finished processing successfully
    - The system is stopping
    - The system has stopped
    - The system is shutting down
    - The system is not initialized
    - The system is down. Test run finished!

    The right side of the screen displays a graphical illustration of the subsystems being used. If an error occurs during the run, *ERR* is displayed on the subsystem that fails. A text message in red indicating a failure is also displayed within the Run Status.

    When errors occur, you can click on the subsystem that created the error to display a description of the error, and to provide more information from which you can research the issue.

To view the status of the current run

You can select the Get Running Workspaces tab to view the run status of the current engine run for a test run or a production run (batch mode only).

1. Select **Engine Manager** from the navigation bar.
2. Select the Get Running Workspaces tab.
3. Select a workspace from the drop-down list.
4. Click **Get Run** to display the status page for the run.
5. To stop the run, click **Stop Run**.

To view the statistical results of a previous engine run

You can view the results of a previous run from the Get Run History tab.

1. Select **Engine Manager** from the navigation bar.
2. Select the Get Run History tab.
3. Only if you have multiple environments configured, select an environment.

4. Select a workspace.
5. Select the run you want to view.

   There may have been multiple runs for the selected workspace. The runs for the selected workspaces are listed by run number and date/time stamp.
6. Click **Get History** to display the history.

   The Run History information and the subsystem graphic give information about that particular run. If an error occurs, you can click on the indicated subsystem to display a description of the error.

Running a production workspace from the command line

You must use a Detect account with Administrator permissions to run a production workspace (pub) from the command line.

Example commands: batch runs

To run the engine from the command line, execute the following statement with the appropriate commands. If you run from a location other than the `Application\bin` directory under your Detect installation, you must enclose the path to the `edmdriver.exe` file in quotes if the path includes any spaces.

```
edmdriver.exe
```

- The following is an example of a command that the Detect administrator John would enter to run the workspace ABC in batch mode for vendor BNK.

  ```
  edmdriver.exe /start /env:2 /dir:c:\Feeds\ABC /r:ABC /v:BNK /entity:a
  /user:John /pswd:password /query:2006-07-25 /arttrans:1
  ```

Example commands: always-on runs

The following are example commands for always-on runs. The commands show the minimum required parameters. Actual commands used in a production environment would probably use some of the optional parameters.

- The following is an example of a command that the Detect administrator Mary would enter to initialize an always-on run of the workspace `FST`.

  ```
  edmdriver.exe -init -r:FST -u:Mary-/p:password -env:1
  ```
- The following is an example of a command that the Detect administrator John would enter to run the engine in always-on mode.

  ```
  edmdriver.exe /run /user:John /pswd:password /env:1
  ```
- Stop an always-on run.

  ```
  edmdriver.exe -stop -u:Mary -p:password -env:1
  ```
- Shut down an always-on run.

  ```
  edmdriver.exe /shutdown /u:John /p:password /env:1
  ```

Command line reference

The following table describes the required and optional commands. Where a command is exclusive to the always-on or batch mode, this is noted in the description. If not noted as exclusive to always-on or batch mode, the command is available in both batch and always-on mode.

The commands are shown with a forward slash (/) to start each command. However, note that you can also use a hyphen (-) to start commands.

Note: If you use rules in your workspace that are based on inactivity, you should check the inactivity mode setting before you run the workspace.

Command	Description
`/arttrans:`*`ATX_code`*	Optional. Specifies whether to create an artificial transaction (ATX), and if so, what type. • 1—Detect does not create an artificial transaction • 2—Detect creates an End of Run transaction • 3—Detect creates an End of Day transaction
`/clearstate`	Optional. Clears entity state before processing. Used only with `/run` (always-on mode) and `/start` (batch mode).
`/dir:`*`folder`*	Specifies the directory path of the data source feeds. This setting is necessary if your configuration does not use multiple cluster units. If you are using multiple cluster units, you do not need this parameter because the location of the data source feeds is set using the Configuration Utility.
`/eetqmode:`*`mode`*	Optional. Specifies the inactivity mode. Valid options are: • `Disabled` • `CRM` • `RealTime` • `RealTime2` See the topic "Task: Adjust system preferences using Advanced Configuration settings" in the "Configuring Multiple Cluster Units" chapter of the *IBM Unica Detect Installation Guide* for descriptions of these options.
`/entity:` *`entity_type_key`*	Optional. Sets the entity type to be processed, identified by an entity type key that is a single, lower-case character code created when Detect was configured, such as `c` for customer. You can enter multiple entities in a comma-separated list. This setting is not necessary if your configuration uses multiple cluster units because if you are, it was set using the Configuration Utility. If you are not using multiple cluster units, you must use this setting with both `start` and `run`.
`/env:`*`environment_ID`*	Required with `start` (batch mode) and `init`, `run`, and `shutdown` (always-on mode). Specifies the environment ID as shown in the Configuration Utility. If the value is invalid or does not reference an existing environment, Detect reports an error and ends the process.
`/feederlog:`*`mode`*	Optional. Sets the feeder log mode. The options are: • `Off`—No trace log is produced. This is the default. • `Trace`—Trace the transactions through the feeder. This is the recommended setting. • `Debug`—Write additional feeder-specific debug information to the log. This setting typically used only at the request of IBM Unica Services.

Command	Description
/ftp:*ftp_flag*	Optional. Specifies whther the engine attempts to execute the batch files for run success, success with errors, or failure. Must be set to On with with start (batch mode) and init and run (always-on mode) if you want run success and run failure files to be created. The options are: • On—Depending upon the outcome of the run, Detect executes one of the files associated with the parameter signaling an end of run. • Off—The process does not execute.
/init	Optional. Initializes the Detect engine, in always-on mode only. Must be used with the following: • /P or /pswd • /r or /W • /U or /user • /env
/log:*mode*	Optional. Sets the trace log mode. The options are: • Off—No log file is produced. This is the default. • Trace—Trace the transactions through the back engine and trigger systems. This is the recommended setting. • Debug—Write additional engine-specific debug information to the log. This setting typically used only at the request of IBM Unica Services.
/pswd: *password*	Required with /start (batch mode) and /init, /run, /stop, and /shutdown (always-on mode). Specifies a password for authentication purposes. Must be the password of the Detect system administrator. An alternative name for this command was introduced in version 8.2.0. You may use /P:*password* for this command.
/query:*date*	Optional. Specifies the inactivity end date for the forward-looking inactivity events that need to be triggered. Use this format: YYYY-MM-DD.
/r:*workspace*	Required with run and start. The 3-character workspace or runspace code. The workspace pub is the correct value for production. This value is case-sensitive. An alternative name for this command was introduced in version 8.2.0. You may use /W:*workspace* for this command.
/recover	Optional. Causes the engine to run in recovery mode.
/run	Runs the Detect engine, in always-on mode only (not batch mode), after the system is initialized using /init. Must be used with the following: • /P or /pswd • /U or /user • /env
/servername:\\ *server_name*	Specifies the name of the server on which the EMC is installed. Used only when you run the engine remotely.

Command	Description
`/shutdown`	Shuts down a run, in always-on mode only (not batch mode). To re-start an always-on run after shutting down, you must initialize and then run. Must be used with the following: • `/env` • `/P` or `/pswd` • `/U` or `/user`
`/start`	Runs the Detect engine, in batch mode only (not always-on). Must be used with the following: • `/P` or `/pswd` • `/r` or `/W` • `/U` or `/user` • `/env` Also,`/dir` is required if you are not using multiple cluster units.
`/stop`	Stops a run, in always-on mode only (not batch mode). After the always-on run stops, always-on remains initialized and may be re-started using the `/run` command. Must be used with the following: • `/env` • `/P` or `/pswd` • `/U` or `/user`
`/user:` *`user_name`*	Required with `/start` (batch mode) and `/init`, `/run`, `/stop`, and `/shutdown` (always-on mode).Specifies a user for authentication purposes. The user must be a Detect system administrator. An alternative name for this command was introduced in version 8.2.0. You may use `/U:`*`user_name`* for this command.
`/v:` *`vendor`*	Specifies a 3-character vendor ID.

Chapter 13. Running and Viewing Reports

The IBM Unica Detect reports allow you to monitor the impact and effectiveness of trigger systems, to enable you to manage the system effectively. You run the standard Detect reports in the Reports Manager.

There are two types of reports: Business Reports and Operational Reports.

About Business Reports

These reports track the impact and effectiveness of the Detect recommended actions, including organizational efficiencies and return on investment (ROI). These reports target those who administer the Detect runsets, workspaces, and components, as well as the business analysts who may never create Detect components. Business impact reports seek to answer the following questions:

- Per run, how many outcomes am I producing?
- What is the volume of outcomes based on distribution channel?
- How many action recommendations are being acted on (i.e. is contact being made with the customer)?
- Are customers responding to the recommended actions?
- What is the overall financial impact of customer responses?
- How can I improve and optimize my components and trigger systems?

About Transaction Volume Reports

The Transaction Volume report provides statistics for the number of transactions per file, that were processed over a selected time period. Transaction counts display in both a graph format and as a statistical list. And, the overall time period is refined by a selected **Time Resolution** that groups the transaction counts into meaningful units for comparison purposes.

To run a Transaction Volume report

1. Click **Transaction Volume** from the left of the interface, in the tree-structure, under **Business Reports** to display the Transaction Volume By File page on the right side of the Reports page:
2. Use the calendar buttons to select the **Start Date** and the **End Date**.
3. Select the **Time Resolution** from the drop-down list.
4. Select the **Runspace Name** from the drop-down list.

   The Runspace list shows the runsets and workspaces you can choose from.
5. Select the **Entity Type** from the drop-down list.
6. Check the **Show Graph** box if you want to display the report results in a graph format.
7. Click **Run Report**.

About Action Volume Totals Reports

The Action Volume Totals report displays information relative to the firing of Action components, for a selected Entity type, within a selected time period. Report results are presented in both a graph format and as a statistical list for each Action component that fired within a selected Runspace.

To run an Action Volume Totals report

1. Click **Action Volume Totals** from the left of the interface, in the tree-structure, under **Business Reports** to display the Action Volume Totals page on the right side of the Reports page:
2. Use the calendar buttons to select the **Start Date** and the **End Date**.
3. Select the **Runspace Name** from the drop-down list.

   The Runspace list shows the runsets and workspaces you can choose from.
4. Check the **Show Graph** box to display the report results in a graph format.
5. Click **Run Report**. The following figure shows an Action Volume Totals report:

About Action Volume Trend Reports

The Action Volume Trend report displays statistics relative to the firing of Action components, within a selected time period, refined and grouped by a Time Resolution. The results display in both a graph format and as a statistical list. The time resolution refinement groups the results into meaningful units for comparison purposes.

To run an Action Volume Trend report

1. Click **Action Volume Trend** from the left of the interface, in the tree-structure, under **Business Reports** to display the Action Volume Trend pane on the right side of the Reports page.
2. Use the calendar buttons to select the **Start Date** and the **End Date**.
3. Select the **Time Resolution** from the drop-down list.
4. Select the **Runspace** from the drop-down list.

   The Runspace list shows the runsets and workspaces you can choose from. Select the actions within this runspace (workspace) that you wish to display.
5. Check the **Show Graph** box to display the report results in a graph format.
6. Click **Run Report**.

About Response Rate Totals Reports

The Response Rate Totals report shows contact and response statistics for the Action components that have fired during a selected time period. Detect is fed as a feed the relevant response data, either as a separate response feed or as a part of the existing feeds. (For example, if the action was drop in debit card usage, then the response becomes an action you create from the same debit card transaction feed, which looks for a resumption in high debit card usage.) For each Action that has fired, the report displays the number of times it has fired, the number of contacts made as a result of the Action, the contact rate, the number of responses as a result of the contact, and the contact rate. Results are presented as a list, and optionally, in a graph format.

To run a Response Rate Totals report

1. Click **Response Rate Totals** from the left of the interface, in the tree-structure, under **Business Reports** to display the Action Volume Totals page on the right side of the Reports page:
2. Use the calendar buttons to select the **Start Date** and the **End Date**.
3. Select the **Runspace Name** from the drop-down list.

   The Runspace list shows the runsets and workspaces you can choose from.
4. Check the **Show Graph** box to display the report results in a graph format.
5. Click **Run Report**.

6. After the report is run, if you want to refine the report to display only a selected list of Actions:
   a. Select the actions that you want to display. You can select multiple actions from the drop-down list using **CTRL** and **SHIFT**.
   b. Click **Run Report**.

About Response Rate Financials Reports

The Response Rate Financials report shows the response return on investment for the Action components that have fired during a selected time period. Detect is fed as a feed the relevant response data, either as a separate response feed or as a part of the existing feeds. (For example, if the action was drop in debit card usage, then the response becomes an action you create from the same debit card transaction feed, which looks for a resumption in high debit card usage.) The sum of the response return on investment for each Action that has fired displays as a list, and optionally in a graph format.

To run a Response Rate Financials report

1. Click **Response Rate Financials Report** from the left of the interface, in the tree-structure, under **Business Reports** to display the Action Volume Totals page on the right side of the Reports page:
2. Use the calendar buttons to select the **Start Date** and the **End Date**.
3. Select the **Runspace Name** from the drop-down list.

   The Runspace list shows the runsets and workspaces you can choose from.
4. Check the **Show Graph** box to display the report results in a graph format.
5. Click **Run Report**.
6. After the report runs, you can refine the report to display only a selected list of Actions using these steps:
   a. Select the actions that you want to display. You can select multiple actions from the drop-down list using **CTRL** and **SHIFT**.
   b. Click **Run Report**.

About Component Firing Trend Reports

The Component Firing Trend report provides statistics on the firing of a selected component over a selected time period. A Time Resolution parameter summarizes the data into relevant groupings. And, the results are displayed in a graph format and as a list. The report is helpful in determining trends in component firings and in evaluating the effectiveness of specific trigger systems.

This report can also be used in troubleshooting, when an action is not firing, to allow you to look further upstream in the trigger logic to see whether components feeding into the low firing action are the ones firing properly.

To run a Component Firing Trend report

1. Click **Component Firing Trend Report** from the left of the interface, in the tree-structure, under **Business Reports** to display the Action Volume Totals page on the right side of the Reports page:
2. Use the calendar buttons to select the **Start Date** and the **End Date**.
3. Select the **Time Resolution** from the drop-down list.
4. Select the **Runspace Name** from the drop-down list.

   The Runspace list shows the runsets and workspaces you can choose from.
5. Select the **Component Type** from the drop-down list.

6. Select an **Event** from the drop-down list. This is the actual component on which you are requesting statistics.
7. Click **Run Report**. The following figure shows a Component Firing Trend report:

About Operational Reports

The target audience for the Operational Reports are those who administer the Detect runsets and components. These reports seek to answer the following questions:

- For a given trigger system, which components are firing and which are not?
- Is my new component or trigger system going to overlap or conflict with one already in existence?
- For a given entity ID, what data is currently saved in the Containers and in state history relative to the completion of components and trigger systems?
- What is the historical volume through the system?
- Where are the performance bottlenecks?

About Individual Trigger Analysis Reports

The Individual Trigger Analysis report offers a visual image of the hierarchy structure of a selected trigger system along with the firing statistics for each component of the selected trigger system. The statistics are presented as both a list of data, as well as in bar graph format.

Depending on the structure of the trigger, these statistics can be helpful in identifying components that are bottlenecks which cause the number of firings of the end action to be lower than desired. Once identified, the criteria on such bottleneck components can be loosened up in order to bring the overall action firing rate nearer to the desired level.

To run an Individual Trigger Analysis report

1. Click **Individual Trigger Analysis** from the left of the interface, in the tree-structure, under **Operational Reports > Trigger Flows** to display the Individual Trigger Analysis page on the right side of the Reports page:
2. Use the calendar buttons to select the **Start Date** and the **End Date**.
3. Select the **Runspace Name** from the drop-down list.

   The Runspace list shows the runsets and workspaces you can choose from.
4. Select an Action from the drop-down list. The list identifies the top-level event Components in the selected runset or workspace.
5. Click **Run Report**.

About the Export Trigger System to Visio report

The Export Trigger System to Visio report provides a visual representation of an Detect trigger system and all of the logic leading to this action trigger in the hierarchy. The report allows you to export trigger logic to a file that can be viewed as a diagram in Microsoft Visio.

Note: You can run the Export Trigger System to Visio report only on trigger systems that result in an action.

When and why to use the report

Use Export Trigger System to Visio report to:

- **Document trigger systems —** Produce visual documentation of the logic in a trigger system after the trigger system has been entered into Detect.
- **Create audit history** — Export trigger diagrams for change control, and track versions. Users can maintain an audit history of the changes made to a trigger by using this feature to record every change. Users can also archive previous versions of the trigger.
- **Troubleshoot** — By obtaining a graphic view of the trigger, users can more easily trace the flow of information through the system and possibly identify problems.
- **Design the flow of triggers** – The report provides a stencil that you can use with Visio as you are designing triggers. The stencil has shapes that represent trigger components. After you create a trigger system, you can run the report and compare the output with the original design.
- **Collaborate —** Collaborate on trigger design and troubleshooting. Users can email the report to colleagues, who can help in troubleshooting and design. (Note that other collaborators who want to add shapes or connectors to the report to help refine the design drawing must have the stencil.)

Software requirements

- Microsoft Visio 2003 or 2007–Visio is not bundled with Detect. The Detect server does not need Visio in order to run this report. However, Visio must be installed on the client machines where the report will be opened.
- Stencil–The stencil provided with Detect is not required in order to open the report. However, it must be on the local machine if you want to add shapes to the report in Visio.
- Pop-ups must be disabled.

About running the Export Trigger System to Visio report

When you run the report, it is saved as a Visio `.vdx` file, which is an XML representation of a Visio diagram.

About saving reports locally with the stencil embedded: You can embed stencil in the report, which allows you to add Detect stencil shapes and connectors to the drawing. The stencil is automatically embedded in reports that are saved to the server. If you want to save reports to your local machine and have the stencil embedded in them, you must download the stencil to your local machine. Then when you run the report, you will specify the stencil filename and location.

To run the report:
This section describes how to run a report.

1. If you want the stencil embedded in it, then the stencil must exist on your local machine. If its not already on your local machine, you can download it using the **Download Stencils** button.
2. In Detect, select **Reports > Export Trigger System to Visio**.
3. Select the **Runspace Name** from the drop-down list.

   This list includes runsets and workspaces.
4. Select the component for which you want the report from the **Component** drop-down list of top level components.

   The drop-down list displays a list of the top-level event components of the Trigger Systems in the selected runspace.
5. In the **Report File Name** textbox, give the file a name or accept the default.

Note: If there is an existing report for the selected component already saved on the server, consider giving the report file a new name. For example, add a number to the filename. Otherwise, the existing report will be overwritten.

The default name of the report depends on whether the component is in a runset or a workspace.

- If the component is in a runset, the name will be based on the runspace code and the name of the top level component.
- If the component is in a workspace, the name will be the workspace code with the name of the top level component.

6. If you want to save a copy of the report to your local machine with the stencil embedded in it, click the folder icon and specify the location of the stencil on your local machine. Otherwise, skip to step 7.
7. Click **Run Report**.

   When you run the report, it is automatically saved to the server in the `*Vendor ID*\Reports` directory under your Detect installation.

   If Detect is installed in the default installation folder, the path begins with `C:\Program Files\Unica`.

   A message appears, informing you that the report generation was successful and allowing you to choose to download the report to your local machine.
8. Click **OK** to save the report locally. Otherwise, click **Cancel**.

   Note: Clicking **OK** also gives you the option to view the report now.

   If you save the report to your local machine, provide a descriptive name for the directory (for example, "My Detect Visio Reports").

About working with a report inside Visio: There are several reasons you might want to work with the Export Trigger System to Visio report within Visio.

- to understand the report flow
- to simplify the report
- to view summary information for a component
- to design trigger systems

Understanding the report flow: The report generally lays out the components on the page so that event flow is easy to read. It separates event generating components from non-event generating components by grouping the auxiliary components (such as Math, Select, and Qualifier) along the top of the report and grouping event generating components (such as Simple, Container Manipulator, and Pattern) along the bottom with the event flow. The components and connectors in the report have distinct shapes, which are provided by the stencil that is installed with Detect.

To simplify the report:

Export Trigger System to Visio reports can be complex. Because the most important thing to examine in the report is the event flow, and you may want to simplify the drawing or reorganize it so that it is easier to focus on the flow.

- **Layering**: Built-in layering of the Visio diagram enables you to selectively hide some of the complexity of the trigger logic. For example, you might focus on an event type layer, remove comments, etc. To do so within Visio, click **View > Layer Properties** and choose which types of components and connectors you would like to display, and which you want to hide.

- **Rearranging connectors**: Lines can overlap on the diagram. You can click on a line to see the connector's ends, and then drag the connector to another position that helps you see the flow.
- **Rearranging shapes**: You can simplify the report by rearranging the placement of the shapes using a simple drag-and-drop method.

To view summary information for a component:
Each of the shapes is labeled with the name of the component. You can get more detailed information about the component:

- Using Visio 2003, the information is available in the Custom Properties window. Right click on a shape in Visio diagram and select **View > Custom Properties Window**.
- Using Visio 2007, the information is available in the Shape Data window. Right click on a shape in Visio diagram and select **View > Shape Data Window**.

The window shows only field information if the data exists for the component. Possible fields are Rule ID, Description, Rule Type, as well as the names of any labels. If the window is too narrow, you can right click on it and select **AutoFit**.

Suggested workflow for designing triggers

You can use the Export Trigger System to Visio report in conjunction with the included stencil to design trigger systems.

1. On paper, sketch a rough outline of the trigger system's main event flow.
2. Create a diagram of the logic of the trigger system using Visio. You can use the shapes in the provided stencil to represent components and connectors.
3. Using Detect, build the trigger system, referring to the Visio diagram you created in step 2.

   Note: You can run the Export Trigger System to Visio report only on trigger systems that result in an action.
4. Run the Export Trigger System to Visio report to generate a .vdx file.
5. Open the file in Visio to see the diagram.
6. Assess the logic and compare the result to the original drawing.
7. Simplify and rearrange the report as needed to make it easier to read.
8. Within Detect, make changes in the trigger system. Add components as needed, and delete the ones that do not work.
9. Repeat steps 4 through 8 as needed.

About the stencil

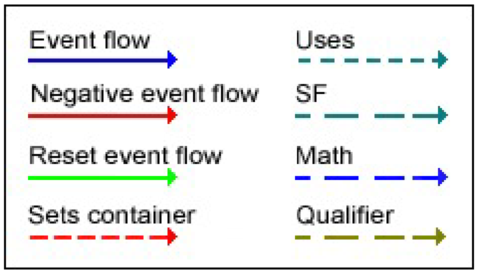
Detect includes a stencil for use with Visio. The stencil provides shapes and lines that correspond with components and connections between components.

About stencil shapes:

- Shapes in the stencil (and on the report) correspond to Detect components. The shapes are identified on the stencil.
- The color of the stencil shapes have meanings
  - Grey – component does not generate an event
  - Turquoise – component generates an event, takes an event, or both
  - Blue – a comment
- The stencil provides special connectors for connecting different types of components.

About stencil connectors: The stencil includes special connectors for connecting different types of Detect components.

Use this connector key to help you to trace the logic of the diagram. For example, the event flow connector indicates the main event flow for the trigger system.



Uses for the stencil: You can use the stencil when you are designing a trigger system, so that you have a set of meaningful shapes with which to work. You can also use the stencil to add shapes or connectors to a report and turn it into a design drawing.

Note: The changes you make to the report are for design purposes only and cannot be imported back into Detect.

To find and deploy the stencil:
You need the stencil if you want to use its shapes as you layout the design of a trigger system or if you want to add shapes to an existing report to help you plan the changes you want to make to the trigger system.

You can access the stencil using either of these methods:

- From the user interface for the Export Trigger System to Visio Report: When you are logged into Detect, you can save the stencil to your local machine. Select **Reports > Export Trigger System to Visio**.

  Click**Download Stencils**. You can open the stencil or save it to any location, such as to a `...\My Documents\My Shapes` folder on your desktop.
- On a server where Detect is installed: The stencil is automatically installed with Detect. The stencil is named `DetectTriggerBuildingComponents.vsx`. It is located in the `Application\Reporting\Visio` directory under your Detect installation.

  If Detect is installed in the default installation folder, the path will begin with `C:\Program Files\Unica`.

About logging for the Export Trigger System to Visio report

The Export Trigger System to Visio report (only) has a pluggable and configurable logging capability that uses Microsoft Enterprise Library Logging Application Block. It enables the report to log more information.

This section describes the key features provided with this logging mechanism, as well as how to configure them.

The log files for the Export Trigger System to Visio report: Logging can be done to different destinations that can be configured. By default, each category listed above goes to its own log file with file names as follows:

- Audit goes to `Application.log`
- Database goes to `Database.log`
- Error goes to `Error.log`

About logging with different categories: The logging categories are:

- Audit – all informational logs
- Database – all database queries
- Error – error messages, including stack trace when an exception is thrown

About the log directory: By default, all log files can be found in `<*DetectInstallDir*>\Application\log`

The location of log files can be changed in the web.config file by changing the FileName attribute of its listener.

About turning logging on and off: Entire Logging or Logging for specific category can be turned on and off through the `web.config` (XML file) at a production machine as needed. For example, in the web.config file in `<DetectInstallDir>\Application`, there are two filters specified:

```
<logFilters>
 <add name="LogEnabled Filter" enabled="true" .../>
 <add name="Category Filter"
  categoryFilterMode="DenyAllExceptAllowed"
  <categoryFilters>
   <add name="Error" />
  </categoryFilters>
 </add>
</logFilters>
```

In this code:

- `"LogEnabled Filter"` turns entire Logging capability on or off by setting its **enabled** attribute `"true"` or `"false"`.
- `"Category Filter"` is a list of Categories that are being logged. By default, only `"Error"` is turned on so that it will not hamper the performance of production machine. But if an error occurs and the information in Error.log is not enough to diagnose the problem, you can turn on `"Database"` and `"Audit"` by adding following lines in `web.config` right after `"<add name="Error" />"`:

  `<add name="Database" />`

  `<add name="Audit" />`

Note: Detect must be stopped and restarted before these changes will take effect.

About log file rolling: Log file rolling is a capability that prevents a log file from becoming too large. By default, all log files roll on daily based and log file name is suffixed by the date. The interval can be changed to week, month, and so on by setting `"rollInterval"` of the listener.

About the Trigger System Detail Report

The Trigger System Detail report provides an overall listing of an entire trigger system. It can be used as a way of documenting triggers. Based on the user selected options, the report includes any or all of the following details.

- Component ID
- Component Type
- Start Date
- End Date
- Modified Date
- Summary

- Description
- Author
- Trigger Level

A user can request a report of a single trigger system or all trigger systems within a selected runspace. Report results for a single trigger system are displayed on the Reports screen, and can be optionally printed or exported to a file. However, when a report of all trigger systems in the selected runspace is requested, the results must be exported to a file.

To run a Trigger System Detail report

1. Click **Trigger System Detail** from the left of the interface, in the tree-structure, under **Operational Reports>Trigger Flows** to display the Trigger System Detail page on the right side of the Reports page:
2. Select the **Runspace Name** from the drop-down list.
3. Select a single **Component** from the drop-down list. The list offers a selection of the top-level components for each Trigger System in the selected runspace.

   Note: You can also select **All** from the drop-down list to select all the components. When you select **All**, you must export the results to a file.
4. Check the selections to display for the trigger system details that you want to show on the report. When none of the checkboxes are selected, the report results display only the component name and hierarchy relationship within the selected trigger system.
5. Click **Run Report**.

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

About the System Run Time Distribution Report

The System Run Time Distribution report provides graphical view of a specific run and the amount of time each process used during the run. It can assist you in optimizing feeder settings, the number of engines running on each server, and therefore overall performance.

The processes displayed are:
- Front Engine TX Processing
- Back EngineTX Processing
- Feeder Manager TX Processing
- Listener Manager TX Processing

To run a System Run Time Distribution report

1. Click **System Run Time Distribution** from the left of the interface, in the tree-structure, under **Performance** to display the System Run Time Distribution page on the right side of the Reports page:
2. Select the **Runspace Name** from the drop-down list.
3. Select a single **Run** from the drop-down list. The list offers a selection of IDs and timestamps for the runs that were processed based on the selected runspace.
4. Click **Run Report**.

About the State Snapshot Report

The State Snapshot produces a report of the information stored in proprietary binary format (state history) for a list of specific Entity IDs. It displays all components for which state history has retained state. For each requested Entity ID the report displays the component type and ID, component parameters and duration, trigger limits, subcomponents names and IDs required to satisfy the conditions of firing, whether those subcomponents have fired, and the timestamp of the required subcomponents that have fired.

To run the State Snapshot report from the Reports Manager

1. Click **State Snapshot** from the left of the interface, in the tree-structure, under **State** to display the State Snapshot page.
2. Select the **Runspace Name** from the drop-down list.
3. Select the **Entity Type** from the drop-down list.
4. For **Entity Selection**, choose an option:
   - Single – Select this option if you want to run the report for a single Entity ID. You can print the report or export it to a file.
   - **All in state table –** Select this option if you want to run the report for all Entity IDs in the state table. The report will output the state for all the IDs in the state table into a file in XML format. You can rename the output file or use the default.
   - **Input file with Entity IDs**–Select this option if you have Entity IDs in a file (one per line). The report will output the state for all the IDs into a file in XML format. You must specify the names for the input file and the output file.
5. If you selected Single in the previous step, enter the **Entity ID**. This is a value for the data source field defined as the **Entity ID** field app type. For example, if your entity is *a* accounts, your entity ID is most likely an account number. Only one ID can be entered.
6. Check the checkboxes in front of the state information that you want to include on the report.
7. Click **Run Report**.

If you ran the report for multiple entity IDs, you have the option of opening the XML file or saving it file to your local machine.

If you ran the report for a single entity ID, the results display on the report page.

About the Trigger System Entity Validation report

The Trigger System Entity Validation reports on trigger system integrity and lists the components in the runspace as well as their associated entities. The report is used in multiple-entity environments to assure that a given trigger system can only access data from a single state object at a single entity level. For example, if there is a container holding all the check amounts for a customer, filled on transactions at the customer level, a trigger at the account level cannot query how many transactions are in that container, because it relates to a customer, not an account.

Detect triggers will therefore not fire correctly if they span multiple entities. If the trigger system contains components that refer to more than one entity type, the entity type will be labeled as Inconsistent.

To run a Trigger System Entity Validation report

1. Click **Trigger System Entity Validation** from the left of the interface, in the tree-structure, under **Operational Reports** to display the Trigger System Entity Validation page on the right side of the Reports page:
2. Select the runspace from the drop-down list.
3. Click **Run Report**.
4. This sample shows one section expanded. You can click the arrow buttons to expand and collapse the sections of the report.

   Note: If **Inconsistent** is given as an Entity Type Name, then the trigger has a component referencing data sources that belong to more than one entity.

About printing and exporting reports

Reports that contain a **Print** and an **Export** button allow you to print and export the contents of the report.

To print a report, click **Print**.

To export a report

1. Click **Export**.
2. Choose an export file format from the drop-down list.

   The available formats are MS Excel (XLS), MS Word (DOC), and Portable Document (PDF).
3. Type the name of the file that the report will be exported to in the text box.
4. Click **Export Now** to export the report.

Chapter 14. Monitoring the Effectiveness of Trigger Systems

The IBM Unica Detect Response manager allows you to monitor the effectiveness of trigger systems. It allows you to correlate the firing of a specific Action component with the expected firing of a response (Action component), based on the firing of an optional contact (Action component), within a projected time frame. The completed cycle, a response behavior, is assigned a monetary value which can be used in calculating return on investment (ROI).

The correlation of the firing of Action components is made via the Response manager.

When Detect processes a run, the system references the saved response behaviors, creating a log of statistics pertaining to the completion state of the behaviors. To view the response statistics in a friendly format, you must then run either the *Response Rate Totals* or *Response Financials* report in the Reports manager.

To use the Response manager user interface

1. Select **Responses** in the navigation bar to display the Label managers. The Response manager displays a tree structure of all the workspaces for the Vendor. This screen is divided into two sections. The right side of the screen stays empty until you choose an operation.
2. Click the expand button (the plus sign) next to the name of the workspace to expand the tree structure and display a list of the response behaviors in the selected workspace.
3. Click one of the already defined response behaviors to display its associated details in the section to the right.

To add a new response behavior

Follow the steps below to add a new response behavior to a workspace.

1. Click **Responses** in the navigation bar to access the Response manager.
2. In the tree structure, click the name of the workspace to which you want to add the new response behavior.

   The right side of the window is updated to show any existing response behaviors. If no response behaviors are defined for the workspace, a message is displayed.
3. While the workspace is selected, click **Add**.

   A blank Response Behavior Details window is displayed.
4. Enter a name for the response behavior in the **Name** field.
5. Select an option from the **Action** drop-down list. (This list is made up of action components in the workspace.)

   It is firing of the selected action that initiates the monitoring of the remainder of these response behavior parameters.
6. Optional: Select a **Contact** from the drop-down list of action components.

   Once the response behavior has been initiated by the firing of the selected action, the Detectengine monitors for the firing of the selected contact.
7. Select a **Response** from the drop-down list of action components.

Once the response behavior has been initiated by the firing of the selected action, the Detectengine monitors for the firing of the selected response.

8. Set the **Quantity** and **Unit** time interval parameters.

   This time interval defines the time period during which the action, contact, and response components must fire for the response behavior to be recognized by the Detect engine.

   The time period starts when the Detect engine recognizes the firing of the action and continues for a length specified by the **Quantity** and **Unit** parameters.
9. Enter a dollar value associated with the successful completion of the response behavior in the **Financial** field.

   This must be a valid dollar amount in the format of *0.00*, with a maximum of seven digits to the left of the decimal point.
10. Check **Each occurrence reinitializes** if you want each firing of the initial action component to re-start the time interval.
11. Click **Save**.

To create a copy of a response behavior

The copy function provides an easy means of creating response behaviors that are based on the same initiating action. When the copy function is used, the Response Behavior Details screen displays the parameters of the original response behavior. By pre-filling these parameters, a user can quickly create similar response behavior variations which can be used to monitor the effectiveness of trigger systems. While the screen allows a user to vary all other parameters, it prohibits changing the initiating "Action".

The screen also offers an option of copying the statistics log of the original response behavior.

1. Click **Responses** in the navigation bar to access the Response manager.
2. In the tree structure, expand the workspace that contains the response behavior you want to copy.
3. Click the name of the response behavior you want to duplicate and click **Copy**.

   The right side of the screen is updated to clear the **Name** field, prevent access to the Action field and introduce additional options for the copy.
4. In the **Name** text box, type a name for the new response behavior.

   Notice that the **Action** parameter since it is greyed, preventing the changing of the initiating Action selection.
5. Optional: Select an Action component from the **Contact** drop-down list.

   Once the response behavior has been initiated by the firing of the selected Action, the Detect Engine monitors for the firing of the selected Contact.
6. Select an Action component from the **Response** drop-down list. Once the response behavior has been initiated by the firing of the selected Action, Detect monitors for the firing of the selected Response.
7. Select a **Time Interval**. This parameter defines the time period during which the Action, Contact, and Response components must fire for the response behavior to be recognized by the Detect Engine. The time period starts when the Detect Engine recognizes the firing of the Action and continues for a length specified by the **Unit** and **Quantity** parameters.
   a. Set the **Quantity** from the drop-down list.

   b. Set the **Unit** from the drop-down list.

8. Select a value that is associated with the successful completion of the response behavior from the **Financial** dropdown. This must be a valid dollar amount in the format of *0.00*, with a maximum of seven digits to the left of the decimal point.
9. Check the **Each occurrence reinitializes** checkbox if you want each firing of the initial Action component to re-start the Time Interval.
10. Choose a copy log option:
    - **Do not copy log—**copies no existing log statistics from the original response behavior. The Detect Engine starts monitoring this new response behavior as if it was created with the **Add** function.
    - **Copy action log only —**copies existing log statistics for the initiating Action component of the original Response behavior. Significant Action firings for the original response behavior are recognized by the Detect Engine when maintaining statistics for the new response behavior copy.
    - **Copy action and contact log—**copies existing log statistics for the initiating Action component and the Contact component of the original Response behavior. Significant Action firings and Contact firings for the original response behavior are recognized by the Detect Engine when maintaining statistics for the new response behavior copy.
11. Click **Save**.

To delete a response behavior

1. Click **Responses** in the navigation bar to display the Response manager screen.
2. In the tree structure, expand the workspace that contains the response behavior you want to delete.
3. Click the response behavior you want to delete.
4. Click **Delete**.
5. Click **OK** to confirm the delete process, or **Cancel** to keep the response behavior.

Appendix A. About the Outcome Table

As Action components fire, IBM Unica Detect inserts outcome rows into the Outcome table.

The Outcome table can reside in any Oracle or SQL Server table to which a connection can be made. While it is traditionally located within the Detect database (which runs on either Oracle or SQL Server), it can reside externally.

The connection to access the Outcome table is set during the Detect installation process. The Configuration Utility lets you view and modify the current connection string.

Detect is delivered with one default outcome table. However, you can create additional outcome tables using the Configuration Utility.

Contents of the Outcome Table

As Actions trigger, outcome rows are inserted into the Outcome table. The Outcome table has the following columns:

Outcome Table Column	Description
Run ID	ID of the run. The Run ID helps distinguish between the outcome of one run versus the outcome of the run before or after it. Because of the Run ID, you do not need to truncate the outcome table after every run because you can associate the outcome with a specific run.
Entity ID	ID of the account, customer, household, site, etc., for which the trigger system fired.
Entity Type	Single character entity type code, such as "a", "c", "h", "s", etc.
Component ID	ID of the Action component that fired. (Also Rule ID.)
Outcome Message	XML format of the text message associated with the Action component. (Most information is in this column.) This is the message that was typed in the text box when the component was created, plus the XML representation of any of the fields that were selected for inclusion in the outcome on the Outcome table of the associated Action component. Dates in the outcome message are formatted according to the then it will be formatted according to the locale setting found in the Configuration Utility's Advanced Configuration settings.
Outcome ID	A numerical, primary key field. The value is set by a trigger upon insert.
Firing Time	The date/time stamp at the time the final event occurred that caused the Action to fire

Outcome Message XML

The XML in the Outcome Message column of the Outcome table can consist of data from a number of different Detect sources. At a minimum it contains a text message that a user typed in the text box when an Action component was created. It can also contain data from the Profile feed file, data from Containers, data from Select functions, and data from Join functions, causing the XML to become quite complex.

The message that was typed in the text box is "this message includes the sum of the values in the All Transactions container and the value in the zipcode field on the Customer Profile data source". The Action component also included the noted values as part of the outcome.

The XML in the Outcome Message appears as follows:

```
<OUTPUT>
   <TEXT>
   this message includes the sum of the values in the All Transactions container
   and the value in the zipcode field on the Customer Profile data source:
   </TEXT>
   <CONTAINER name="All Transactions" function="SUM">
   125
   </CONTAINER>
   <DATASOURCE name="Customer Profile" field="zipcode">
   11746
   </DATASOURCE>
   </OUTPUT>
```

Data types in outcome messages

The possible data types in Outcome Messages are listed below:

Table 11. Data Source: Profile Feed File

Type of Data	Data Selection Function	Value of Data
Field	(none)	The value contained in the field.

Table 12. Data Source: Container

Type of Data	Data Selection Function	Value of Data
Total Container contents. Represented by "*"	Count	An integer representing the number of entries in the Container.
Total Container contents. Represented by "*"	Data	The actual data values of every field, in every row in the Container. The number of fields, name of fields, and order of fields in each row is determined by the Type Descriptor defined in the Container.
Field	Average	The average of the values in the Container for the specified field.
Field	Count	An integer representing the number of entries in the Container for the specified field.
Field	Count Distinct	An integer representing the number of unique entries in the Container for the specified field.
Field	Maximum	The actual data value of the highest of the values saved in the Container for the specified field.

Table 12. Data Source: Container (continued)

Type of Data	Data Selection Function	Value of Data
Field	Minimum	The actual data value of the lowest of the values saved in the Container for the specified field.
Field	Mode	The actual data value of the most frequently occurring value saved in the Container for the specified field.
Field	Sum	An integer representing a total of the values in the Container for the specified field.
Field	Standard Deviation	A numeric value representing the measure of the scatter of values around the average of the data values saved in the Container for the specified field.
Field	Data	The actual data values for the specified field, in every row in the Container.

Table 13. Data Source: Select Function

Type of Data	Data Selection Function	Value of Data
All selected rows. Represented by "*"	Count	An integer representing the number of rows returned by the Select function.
All selected rows. Represented by "*"	Data	The actual data values of every field, in every row returned by the Select Function. The number of fields, name of fields, and order of fields in each row is determined by the Type Descriptor defined in the Select Function.
Field	Average	The average of the values returned by the Select Function for the specified field.
Field	Count	An integer representing the number of entries returned by the Select Function for the specified field.
Field	Count Distinct	An integer representing the number of unique entries returned by the Select Function for the specified field.
Field	Maximum	The actual data value of the highest of the values returned by the Select Function for the specified field.
Field	Minimum	The actual data value of the lowest of the values returned by the Select Function for the specified field.
Field	Mode	The actual data value of the most frequently occurring value returned by the Select Function for the specified field.
Field	Sum	An integer representing a total of the values returned by the Select Function for the specified field.
Field	Standard Deviation	A numeric value representing the measure of the scatter of values around the average of the data values returned by the Select Function for the specified field.

Table 13. Data Source: Select Function (continued)

Type of Data	Data Selection Function	Value of Data
Field	Data	The actual data values for the specified field, in every row returned by the Select Function.

Table 14. Data Source: Join Function

Type of Data	Data Selection Function	Value of Data
All selected rows. Represented by "*"	Count	An integer representing the number of rows returned by the Join function.
All selected rows. Represented by "*"	Data	The actual data values of every field, in every row returned by the Join Function. The number of fields, name of fields, and order of fields in each row is determined by the Type Descriptor defined in the Join Function.
Field	Average	The average of the values returned by the Join Function for the specified field.
Field	Count	An integer representing the number of entries returned by the Join Function for the specified field.
Field	Count Distinct	An integer representing the number of unique entries returned by the Join Function for the specified field.
Field	Maximum	The actual data value of the highest of the values returned by the Join Function for the specified field.
Field	Minimum	The actual data value of the lowest of the values returned by the Join Function for the specified field.
Field	Mode	The actual data value of the most frequently occurring value returned by the Join Function for the specified field.
Field	Sum	An integer representing a total of the values returned by the Join Function for the specified field.
Field	Standard Deviation	A numeric value representing the measure of the scatter of values around the average of the data values returned by the Join Function for the specified field.
Field	Data	The actual data values for the specified field, in every row returned by the Join Function.

Format of Outcome Message XML

Realizing that this complexity can make the parsing of the outcome difficult, Detect uses a standardized structure for the XML. The XML groups the data into <DATASOURCE> blocks.

- The first block is always the text message that a user typed in the text box of the Action component.
- The blocks that follow represent the "Additional Information" that a user selected in the Action component.
- The blocks display in the order in which the data was selected.

The XML data blocks use formatting standards for data from text, profile feed file, container, select function, and join function.

Text

The text portion of the output is delimited by the TEXT boundaries, and is always in the following format:

```
<TEXT>actual text message that was typed in the Action</TEXT>
```

Example:

```
<TEXT>testing container dump</TEXT>
```

Profile Feed File

Data from a Profile Feed Files is delimited by the DATASOURCE boundaries.It includes:

- The name of the profile file - displayed in quotation marks
- The name of the field in the profile file - displayed in quotation marks
- The actual value of the field

The format is as follows:

```
<DATASOURCE name="file name" field="field name">
   field value
   </DATASOURCE>
```

Example:

```
<DATASOURCE name="profile" field="age">
   29
   </DATASOURCE>
```

Note: This XML repeats for every profile field that is included in the outcome.

Container

Data from a Container is delimited by the DATASOURCE boundaries. It includes the following:

- The name of the Container - displayed in quotation marks
- A "Function(Field)" designation - displayed in quotation marks
- The actual value returned by the "Function(Field)" designation. It is a single value when the function is Average, Count, Count Distinct, Max, Min, Mode, Sum, or Standard Deviation. It repeats for every row of data in the Container when the function is Data. The field can be either the name of a field in the Container, or * for every field in the Container

When "Function(Field)" returns one value, the format is as follows:

```
<DATASOURCE name="container name" field="FUNCTION(FIELD NAME)">
   actual value
   </DATASOURCE>
```

Example:

```
<DATASOURCE name="TRANSACTION_HISTORY"field="MAX(TRAN_AMT)">9245
</DATASOURCE>
```

When the function is Data, the format is as follows:

```
- <DATASOURCE name="container name" field="DATA(*)">
   - <ROW>
        <FIELD name="TimeStamp">field value</FIELD>
        <FIELD name="FIELD1 NAME">field value</FIELD>
        <FIELD name="FIELD2 NAME">field value</FIELD>
        <FIELD name="FIELD+n NAME">field value</FIELD>
    </ROW>
 - <ROW+n>
        <FIELD name="TimeStamp">field value</FIELD>
        <FIELD name="FIELD1 NAME">field value</FIELD>
        <FIELD name="FIELD2 NAME">field value</FIELD>
        <FIELD name="FIELD+n NAME">field value</FIELD>
    </ROW+n>
 </DATASOURCE>
```

Example:

```
- <DATASOURCE name="TRANSACTION_HISTORY"field="DATA(*)">
  - <ROW>
       <FIELD name="Timestamp">38057</FIELD>
       <FIELD name= "TRAN_AMT">300</FIELD>
       <FIELD name="TRAN_CODE">203</FIELD>
   <ROW>
 - <ROW>
       <FIELD name="Timestamp">38061</FIELD>
       <FIELD name= "TRAN_AMT">355</FIELD>
       <FIELD name="TRAN_CODE">203</FIELD>
   </ROW>
</DATASOURCE>
```

Select Function

Data from a Select Function is delimited by the DATASOURCE boundaries. It includes:

- The name of the Select Function - displayed in quotation marks
- A "Function(Field)" designation - displayed in quotation marks
- The actual value returned by the "Function(Field)" designation It is a single value when the function is Average, Count, Count Distinct, Max, Min, Mode, Sum, or Standard Deviation. It repeats for every row of data in the Select Function, when the function is Data. The field can be either the name of a field in the Select Function, or * for every field in the Select Function

When "Function(Field)" returns one value, the format is as follows:

```
<DATASOURCE name="select function name" field="FUNCTION(FIELD NAME)">
   actual value
   </DATASOURCE>
```

Example:

```
<DATASOURCE name="Sum Tran Amt by Tran Code"field="MAX(TRAN_AMT)">
   2255
   </DATASOURCE>
```

When the function is Data, the format is as follows:

```
    - <DATASOURCE name="select function name" field="DATA(*)">
   - <ROW>
        <FIELD name="FIELD1 NAME">field value</FIELD>
        <FIELD name="FIELD2 NAME">field value</FIELD>
        <FIELD name="FIELD+n NAME">field value</FIELD>
    </ROW>
 - <ROW+n>
        <FIELD name="FIELD1 NAME">field value</FIELD>
        <FIELD name="FIELD2 NAME">field value</FIELD>
        <FIELD name="FIELD+n NAME">field value</FIELD>
    </ROW+n>
 </DATASOURCE>
```

Example:

```
- <DATASOURCE name="Sum Tran Amt by tran Code"field="DATA(*)">
  - <ROW>
   <FIELD name= "TRAN_AMT">2505</FIELD>
       <FIELD name="TRAN_CODE">203</FIELD>
   </ROW>
</DATASOURCE>
```

Join Function

Data from a Join Function is delimited by the DATASOURCE boundaries. It includes:

- The name of the Join Function - displayed in quotation marks
- A "Function(Field)" designation - displayed in quotation marks
- The actual value returned by the "Function(Field)" designation It is a single value when the function is Average, Count, Count Distinct, Max, Min, Mode, Sum, or Standard Deviation. It repeats for every row of data in the Join Function, when the function is Data. The field can be either the name of a field in the Join Function, or * for every field in the Select Function.

When "Function(Field)" returns one value, the format is as follows:

```
<DATASOURCE name="join function name" field="FUNCTION(FIELD NAME)">
   actual value
   </DATASOURCE>
```

Example:

```
<DATASOURCE name="Unique Tran Amt by Tran Code"field="MIN(TRAN_AMT)" >
   2
   </DATASOURCE>
```

When the function is Data, the format is as follows:

```
- <DATASOURCE name="join function name" field="DATA(*)">
   - <ROW>
        <FIELD name="FIELD1 NAME">field value</FIELD>
        <FIELD name="FIELD2 NAME">field value</FIELD>
        <FIELD name="FIELD+n NAME">field value</FIELD>
    </ROW>
 - <ROW+n>
        <FIELD name="FIELD1 NAME">field value</FIELD>
        <FIELD name="FIELD2 NAME">field value</FIELD>
        <FIELD name="FIELD+n NAME">field value</FIELD>
    </ROW+n>
 </DATASOURCE>
```

Example:

```
- <DATASOURCE name="Unique Tran Amt by tran Code"field="DATA(TRAN_CODE)">
  - <ROW>
  <FIELD name="TRAN_CODE">204</FIELD>
   </ROW>
  - <ROW>
  <FIELD name="TRAN_CODE">210</FIELD>
   </ROW>
  - <ROW>
  <FIELD name="TRAN_CODE">423</FIELD>
   </ROW>
</DATASOURCE>
```

Appendix B. G_Run Table

G_Run and its satellite tables are the system journal, providing information on each IBM Unica Detect engine run. The G_Run table contains six columns:

- RUNID — Connects all the information for a specific run to its associated satellite tables.
- VENDORROLEID — The workspace
- STARTTIME — Timestamp for the beginning of a run
- ENDTIME — Timestamp for the completion of a run
- GLOBALSTATE — Indicates the overall stat of the run, 1=down, 2=up
- USERID — The ID of the user that initiates the run

The satellite tables are follows:

- G_RunStatus — Holds system status history for each run
- G_SubsystemStatus — Holds system status history for each run per subsystem
- G_Logs—Holds run logs (only significant log entries)
- G_Errors—Holds run errors
- G_SubsystemPerf—Holds run performance information per batch cycle
- G_RuleFiringCount—Holds sums of rule firing counts per rule per batch

The values for G_RunStatus.GLOBALSTATE are listed below.

0	NOT_IN_MEMORY	For EMC's use, this is also the initial state in DB.
1	UNINITIALIZED	All subsystems are shut down.
2	INITIALIZING	Subsystems are starting up in preparation for a run.
3	<removed>	
4	UP	Subsystems are ready to run after initialization.
6	RUN_PROCESS	Subsystems are actively processing transactions.
7	RUN_END_SUCCESS	All processing is complete with no errors. Subsystems are still up.
8	RUN_FAILED	Processing has failed because one or more subsystems has errored.
9	STOPPED	Subsystems are up but have stopped processing.
10	SHUTTING_DOWN	Subsystems are shutting down.
11	RUN_END_SUCCESS_ERRORS	All processing has completed successfully, but one or more subsystems has encountered errors.
12	STOPPING	Subsystems are in the process of stopping.
13	STOP_FAILED	One or more subsystems encountered an error while stopping.

14	SHUTDOWN_Failed	One or more subsystems encountered an error while shutting down.
15	INIT_FAILED	One or more subsystems has failed to start up.
16	CREATE_FAILED	The TSM could not be created.
17	WAS_NOT_CREATED	The TSM has not been created.
18	DOES_NOT_EXIST	The TSM does not exist.

Appendix C. Grammar Basics of the Engine Trace Log

Note: Information about the IBM Unica Detect trace log grammar is included here for advanced reference purposes.

Each line of the trace log is a tab-separated list of values that represent a single action inside the engine. Every line begins with a tag in all caps that indicates the type of activity performed at that point. Following the tag is a tab separated list of values providing detailed information about the action. The tab is used as the delimiter because it is not currently feasible to include the tab character anywhere in either engine input or output. This format guarantees that any data included in the log will not affect parsing.

Example:

```
TAG value 1 value 2 value 3
```

Grammar description

As shown in the table below, the grammar can be grouped into several areas of concern: General, Events, State, Data, and RHS actions. Each area is listed separately, with definitions for each tag used. Each tag requires a different set of values and thus its own definition in the grammar. Each item for the grammar is defined with the tag in all caps followed by a definition and example. After the definition of the tag, the data elements for that tag are shown in angle brackets <> with a description of each.

General

RUN

- Example: RUN 2004-11-23 11:23 am 1.0
- Description: Used as the first line of the log.

Data Element	Description
<date time>	The current date and time of the system the engine was running on. ISO format (YYYY-MM-DD HH:mm am/pm) will be used to maintain consistency.
<grammar version>	Reports the version of the grammar format. If you specify the grammar version, you can apply changes while maintaining backward compatibility.
<release version>	The release version of Detect on which the log was created.
<binary version>	The version of the engine binary.
<db schema>	The schema code of the Detect database.
<vendor>	The vendor you assign.

Data Element	Description
<role>	The rule set you assign.
<cache mode>	The cache mode used to cache state history.
<cache size>	The size of the cache used to cache state history.
<path>	The local path of the engine binary.
<db connection>	The database connection used during the run.

ENTITY

- Description: Represents the entity processed. Only one entity will appear in the tree view. All other tags are subordinate to this tag.

Data Element	Description
<id>	The entity's unique identifier.
<transaction headers>	A pipe delimited list of the transaction feed field names.
<transaction values>	A pipe delimited list of the transaction field values.

TX

- Example: TX 2004-11-23 12:00 am 1
- Description: Indicates the beginning of a new transaction. A single user can have multiple transactions in a single run. All log entries between this tag and the next TX tag are subordinate to this transaction.

<date-time>	The date-time of this transaction. This date-time is used for all processing on this transaction.
<transaction set mode>	An indicator of the number of transactions left to process in the entire set of transactions for this user. The final transaction will be indicated when this value is equal to 0.
<transaction headers>	A pipe-delimited list of the transaction feed field names.
<transaction values>	A pipe-delimited list of the transaction field values.

STOP

- Example: STOP 123 Made a deposit
- Description: Marks a trouble spot where a trigger system halted its progress. The actual reason for the stoppage will be indicated in another (preceding) log entry. The STOP tag is simply to make it easy to identify all trouble spots at once. The STOP tag has no subordinate elements, but is associated with the previous entry.

Data Element	Description
<ruleid>	The id of the rule that stopped the trigger system.
<rulename>	The name given to the rule by the rule designer.
<reason>	A description of the reason for the STOP.

ERROR

- Description: The ERROR tag reports the occurrence of an error within the engine that halted transaction processing. No details are provided. Information on the error is located in the error logs. An ERROR tag is one of the possible entries that can produce a STOP tag. The ERROR tag is the only tag that will always generate a STOP tag.

DEBUG

- Example: DEBUG engine handleEventQuery Current timestamp previous timestamp. Resetting rule durations.
- Description: A class of entries intended for QA and developers for reporting information useful in debugging. DEBUG entries will not be exposed to end users. A DEBUG entry may appear subordinate to any other tag, but has no subordinates of its own.

Data Element	Description
<class>	The name of the class the entry occurs in.
<method>	The name of the class method the entry occurs in.
<message>	The information message. This element has no set format. It may contain many lines of information.

Events

EVENT

- Example: EVENT 123 0 Made a deposit
- Description: An event about to be matched. The source of the event is not tracked or known. The event may be the first event from the Front Engine, or may be the last in a long chain of events. All log entries between this tag and the next EVENT tag within the current transaction are subordinate to this event.

Data Element	Description
<ruleid>	The id of the rule that generated the event.

Data Element	Description
<subeventid>	An additional characterization of the event. Possible values of the sub-event id are as follows: • 0: This event is a fully matched event from the Front Engine. Only Simple rules may generate this kind of sub-event id. • -1: This event was fired internally by the back engine. All rules capable of generating events, except Simple rules and rules that generate EETQ events may generate this kind of sub event id. • -2: This event was fired from the EETQ. • >0: A positive sub event id will only be associated with Simple rules and indicates a partial match from the Front Engine. The Back Engine will complete the match and generate an event with a -1 sub-event id. • 9: This sub-event id is unique when associated with rules id 0. It represents a Check Queue event from the feeder. This event is the method used to fire all mature EETQ events for the entity.
<rulename>	The name given to the rule by the rule designer.

MATCH

- Example: MATCH 234 Made Three Deposits in a Week
- Description: Indicates a rule that has subscribed to the current event and has been matched. All log entries between this tag and the next MATCH tag within the current event are subordinate to this match.

Data Element	Description
<ruleid>	The rule id of the rule that has matched the current event.
<rulename>	The name given to the rule by the rul.e designer

State

ARM

- Example: ARM 234 Made Three Deposits in a Week
- Description: Indicates that a rule advanced in state without actually firing. The details of the arming would be found in a STATE log entry. The ARM tag has no subordinates.

Data Element	Description
<ruleid>	The id of the rule that armed.
<rulename>	The name given to the rule by the rule designer.

AGE

- Example: AGE 234 Made Three Deposits in a Week 2004-11-01 12:00 AM 112.99|2004-08-09 9:30 AM

- Description: Indicates a rule has lost state because some part or all of it is too old and has gone stale. Details of an aging event would be found in a STATE log entry. An AGE entry is one of the possible entries that can be responsible for a STOP entry. The AGE tag has no subordinates.

Data Element	Description
<ruleid>	The id of the rule that aged.
<rulename>	The name given to the rule by the rule designer.
<cutoff date>	The date, in ISO format, that reports the limit that was crossed by the aged element. The aged element was older than this date.
<aged element>	The element in the rule state that was dropped for being too old.

RESET

- Example: RESET 345 No check use after deposit.
- Indicates a rule whose state was reset. The state of this rule was completely dropped or returned to a base state. The RESET entry is one of the possible entries that can be responsible for a STOP entry. The RESET tag has no subordinates.

Data Element	Description
<ruleid>	The id of the rule that aged.
<rulename>	The name given to the rule by the rule designer.

POST_STATE

Data Element	Description
<ruleid>	No descriptions available.
<rulename>	
<resource>	
<message>	
<fieldlist>	
<valuelist>	

PRE_STATE

- Example: PRE_STATE 345 No check use after deposit FLI
- Description: Reports the current state of a stateful rule in the same manner as the State Snapshot. PRE_STATE reports the state before work was performed and POST_STATE after. The tags are otherwise identical.

Data Element	Description
<ruleid>	The id of the rule.
<rulename>	The name given to the rule by the rule designer.
<resource>	The name of the dynamic resource being reported. The possible values for this value are:

PATTERN
- FLI (forward looking inactivity)
- BLI (backward looking inactivity)

CONTAINER

Data Element	Description
<smessage>	A string of state information to display information that will not be shown in the Display List.
<fieldlist>	A pipe-delimited list of columns names to describe the state information displayed. Each resource uses a different set of columns.

Data

GET
- Example: GET 1 Transaction 10 tx_code TXFEED XX string
- Description: Reports data retrieval from a datasource. There is no knowledge of who has made the request for data, only that one was made. The GET tag has no subordinates.

Data Element	Description
<datasourceid>	The id of the data source.
<datasourcename>	The name given to the data source by the designer.
<fieldid>	The id of the data source field.
<fieldname>	The name of the field given by the designer.
<datasourcetype>	Identifies the type of the data source. The possible values for this value are: • TXFEED: The transaction from the transaction feed. • VPDICT: The customer profile from the profile feed. • DB_TABLE: An external database table. • CONTAINER: An internal container. • SELECT: An internal select query.

Data Element	Description
<value>	The value returned after fulfilling the request
<datatype>	The datatype of the returned value.
<default>	Boolean flag indicating whether or not a default value was returned.

LOGIC

- Example:LOGIC 0 12 IsAGoldCustomer AND true
- Description: Reports the evaluation of a logic expression. Because of the possibility of more than two operands and the difficulty of representing them informatively, only result of the evaluation is reported. The details of evaluation are found in previous entries because all operands of the expression report their own results. The LOGIC expression may have one or more LOGIC or COMP entries as subordinates, but unlike previous tags the subordinates of a LOGIC tag precede it. The LOGIC tag is one of the tags that can be responsible for a STOP tag. In most cases, if the <result> value is "false", and the expression is named, a STOP tag will follow it. There are circumstances where an unnamed expression is part of a larger rule and a false result will generate a STOP entry. There is no circumstance where a "true" result will generate a STOP entry.

Data Element	Description
<id>	The id of the Qualifier rule the logic expression represents. When the logic expression is not the top level expression of a rule, this value will be 0.
<name>	The name given to the Qualifier rule by the designer. When the logic expression is not the top level expression of a rule, this value will be an empty string.
<operation>	The logical operation performed by the expression. The possible values are: • AND • OR • NAND • NOR
<result>	The result of the evaluation. This value will be either "true" or "false."

COMP

- Example: COMP 0 13 IsASeniorCitizen 25 GREATER_THAN_OR_EQUAL 65 false
- Example: COMP 2 0 0 EQUAL 0 true
- Description: Reports the evaluation of a comparison expression. Unlike the LOGIC tag, the values as well as the operation are presented here because of the ease with which this can be done and because it makes syntactical sense. The details of retrieving the values are found in the entries preceding the COMP tag. The COMP tag may have any other data entries as subordinates except for LOGIC and COMP. The COMP tag is one of the tags that can be responsible for a STOP tag. In most cases, if the <result> value is "false", and the expression is

named, a STOP tag will follow it. There are circumstances where an unnamed expression is part of a larger rule and a false result will generate a STOP entry. There is no circumstance where a "true" result will generate a STOP entry.

Data Element	**Description**
<id>	The id of the Qualifier rule the comparison expression represents. When the comparison expression is not the top level expression of a rule, this value will be 0.
<name>	The name given to the Qualifier rule by the designer. When the comparison expression is not the top level expression of a rule, this value will be an empty string.
<value1>	The value of the first operand used in the comparison. The details of retrieving this value precede the COMP tag.
<operation>	The comparison operation performed by the expression. The possible values are: • EQUAL • NOT_EQUAL • GREATER_THAN • LESS_THAN • GREATER_THAN_OR EQUAL • LESS_THAN_OR_EQUAL
<value2>	The value of the second operand used in the comparison. The details of retrieving this value precede the COMP tag.
<result>	The result of the evaluation. This value will be either "true" or "false".

MATH

- Example: MATH 3 101 TotalCost 99.0 TIMES 10 990.0
- Example: MATH 2 0 21 PLUS 1 22
- Description: Reports the evaluation of a math expression. Unlike the LOGIC tag, the values as well as the operation are presented here because of the ease with which this can be done and because it makes syntactical sense. The details of retrieving the values are found in the entries preceding the MATH tag. The MATH tag may have any other data entries as subordinates except for LOGIC and COMP.

Data Element	**Description**
<eid>	The id of the Math rule the expression represents. When the math expression is not the top level expression of a rule, this value will be 0.
<name>	The name given to the rule by the designer. When the math expression is not the top level expression of a rule, this value will be an empty string.
<value1>	The value of the first operand used in the operation. The details of retrieving this value precede the MATH tag.

Data Element	Description
<operation>	The operation performed by the expression. The possible values are: • PLUS • MINUS • TIMES • DIVIDE • MODULUS
<value2>	The value of the second operand used in the operation. The details of retrieving this value precede the MATH tag.
<result>	The result of the evaluation. This value will be either "true" or "false."
<datatype>	The datatype of the result.

SELECT

- Example: SELECT 30 Account Withdrawals CONTAINER [type]=cash 3 rows
- Description: Reports the execution of a select query on a datasource (usually a container). Because selects return a record set result, they will usually be subordinate to some other Data entry. A SELECT tag may have subordinate entries that record the information retrieved for the where clause or the source of the query may be another SELECT or JOIN entry.

Data Element	Description
<datasourceid>	The id of the datasource being queried.
<datasourcename>	The name of the datasource being queried.
<datasourcetype>	Identifies the type the datasource. The possible values for this value are: • TXFEED The transaction from the transaction feed. • VPDICT The customer profile from the profile feed. • DB_TABLE An external database table. • CONTAINER An internal container. • SELECT An internal select query.
<whereclause>	The where clause applied when the query is executed. This value may be an empty string.
<header>	A pipe-delimited list of the field names queried in the select operation.
<rows>	A pipe-delimited list of the values resulting from the query.

JOIN

- Example: JOIN 567 SKU Details 61 Purchased Items 62 Product Table 15 rows
- Description: Records the execution of a join on two data sources. Because the JOIN result is a record set, it will usually be subordinate to some other Data entry. The JOIN tag will have subordinate entries preceding it recording the retrieval of the two data sources it acts upon.

Data Element	Description
<ruleid>	The id of the rule the making the query.
<name>	The name given to the querying rule by the designer.
<parentid>	The datasource id of the parent datasource.
<parentname>	The name of the parent datasource.
<childid>	The id of the child datasource.
<childname>	The name of the child datasource.
<result>	The record set that is the result of the join.

Right Hand Side (RHS) Actions (the "then" part of the rule)

FIRE

- Example: FIRE 234 -1 Made Three Deposits in a Week
- Description: A rule has fired an event. The event has been inserted into the firing list, but is not being processed yet. This same event will appear later in the log under the EVENT tag if nothing stops processing before the event is reached. The FIRE tag has no subordinates.

Data Element	Description
<ruleid>	The id of the rule firing the event.
<subeventid>	An additional characterization of the event. Possible values of the sub event id for a FIRE tag are the same as those for the EVENT tag.
<rulename>	The name given to the rule by the rule designer.

OUTPUT

- Example: OUTPUT 301 Heavy Trader Notice cmdb:Joe Shmoe is a heavy trader. Contact with campaign offer.
- Description: Reports the output generated by a rule. An OUTPUT may have no subordinates, or it may have subordinate DATA reporting the retrieval of information used to build the outcome message. These subordinate entries will precede the OUTPUT tag.

Data Element	Description
<ruleid>	The id of the rule generating the outcome message.
<rulename>	The name given by the rule designer.
<output string>	The message inserted into the outcome data sink.

INSERT

Data Element	Description
<datasourceid>	No descriptions available.
< datasourcename >	
<fieldlist>	
<valuelist>	

SOFT_INSERT

- Example: INSERT 10 Purchases SKU|amount|cost|timestamp 001|1|9.99|2004-11-23 3:55 pm
- Description: Records the insert of data into a container. An INSERT tag will be subordinate only to a MATCH tag, but will have multiple data entry subordinates that report the retrieval of the values to be inserted. Subordinate entries precede the INSERT tag. The information added by a SOFT_INSERT operation will not be committed until the transaction has been completely processed. It is otherwise the same as an INSERT.

Data Element	Description
<datasourceid>	The id of the datasource targeted for the insert.
<datasourcename>	The name of the datasource targeted for the insert.
<fieldlist>	A pipe-delimited list of the fields in the target datasource.
<valuelist>	A pipe-delimited list of the values inserted into the target datasource.

DELETE

- Example: DELETE 202 ClearPurchaseHistory 10 Purchases
- Description: Records the delete of data from a container. The DELETE tag is subordinate to only the MATCH tag, but can have multiple subordinate entries that record the information retrieved for the where clause. These subordinate entries will precede the DELETE tag.

Data Element	Description
<ruleid>	The id of the rule executing the delete.
<rulename>	The name of the rule executing the delete.
<datasourceid>	The id of the datasource targeted for the delete.

Data Element	Description
<datasourcename>	The name of the datasource targeted for the delete.
<whereclause>	The where clause that is applied when the query is executed. This value may be an empty string.

Appendix D. Troubleshooting and Log Files

This appendix describes troubleshooting for IBM Unica Detect.

About troubleshooting a clustered environment

Troubleshooting Detect sometimes involves the analysis of logs. If you are running Detect in a clustered configuration introduces a new dimension complexity because each cluster has its own set of log files.

Detect can be run from the Engine Manager UI and from the command line; the strategy for troubleshooting is different for each method.

About running from the Engine Manager UI

The Engine Manager provides a way of analyzing faults without having to open the log files and thereby offers the most convenient way to troubleshoot. The UI shows a dynamic diagram of the running system and if one of the subsystems fails, on any of the clusters, its image displays the word "ERR". Click on the subsystem in error and a separate window displays the error messages. These messages often give a good indication of the problem; it should not be necessary to consider any of the cluster specifics in order to correct the problem. Some examples of these types of errors include:

- The wrong data sources were used. Because you must specify the entity type when running the engine, you may have used files of the wrong entity type.
- One of the clusters points to a wrong path. Because each of the clusters has its own feed file directory, one of those directories may have been wrong.
- Hardware or network error. One of the clusters may have lost connectivity.
- Database error: The database may be down or that the connection string to the database may be incorrect.

You will have to diagnose other types of errors by referring to the log files on the individual clusters. This activity is identical to the process of troubleshooting the application when running from the command line.

About running from the command line

Production runs of Detect are usually done in conjunction with a scheduler using the command line interface. In this case Detect indicates failure by invoking a batch file and the only way to detect the error is by analyzing the log files.

About Detect log files

Each server that runs Detect has log files. Most of the log files are in a `*Vendor*\logs` directory under your Detect installation.

Point the engine trace utility here for the debug logs. In this path, *`Vendor`* is the three-character name of the vendor as specified during installation.

Log files also exist in the `Affinium Detect\EMC` directory under the IIS web server installation.

About the most informative log files

Detect produces several log files. This section describes the log files that provide the most information and where to find them.

Important: At each startup, the log files are overwritten. If you want to save them, use the Archive Service.

Note: If there are two or more error log files, the file with the most recent timestamp is the most current.

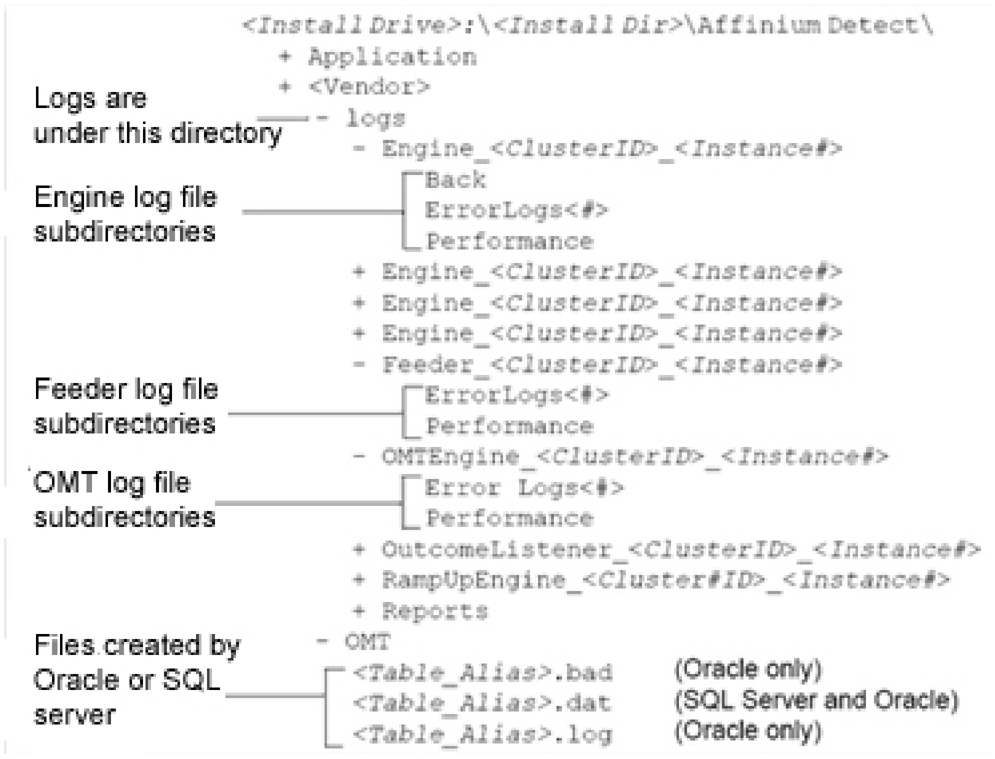
- **TSM log –** There is only one TSM (Trigger Set Manager) log per system configuration, and it resides on the web server unit. It is the most important log, and it gathers information about all of the Detect components. It is located in:
- **Engine logs –** Engine logs are located on each cluster unit in the directory under your Detect installation. `*vendor* \logs\Engine_*ClusterId_InstanceNo*`. There are multiple engine directories if the cluster is running multiple instances of the engine. Each copy of the engine is reflected in a different *InstanceNo*. For example, if there are three engines defined for a cluster with a ClusterID of 4, the log files would look like this: Engine_4_1, Engine_4_2, and Engine_4_3.

  An engine folder has a subdirectory called Back if tracing (trace utility) or debugging is turned on in the engine.
- **OutcomeListener logs** and **Feeder Logs –** Each cluster unit contains an outcome listener log and a feeder error log, contained in *vendor*\OutcomeListener_*ClusterId*_1\ErrorLogs1 and *vendor*\Feeder_*ClusterId*_1\ErrorLogs1, respectively.
- **EMC log –** There is only one EMC log and it resides on the web server unit in *vendor*\EMC\ErrorLog1. This log is useful in the rare case that the failure occurs before the system starts. You may refer to the server configuration in the administrative UI or the Configuration Utility to find it.

Log file directory example

These examples show the log file directory structure for the cluster unit and the web server unit when they are on separate servers.

Sample directory structure on the cluster unit


Logs are under this directory
Engine log file subdirectories
Feeder log file subdirectories
OMT log file subdirectories
Files created by Oracle or SQL server
(Oracle only)
(SQL Server and Oracle)
(Oracle only)

```
<Install Drive>:\<Install Dir>\Affinium Detect\
  + Application
  + <Vendor>
  - logs
    - Engine_<ClusterID>_<Instance#>
        Back
        ErrorLogs<#>
        Performance
    + Engine_<ClusterID>_<Instance#>
    + Engine_<ClusterID>_<Instance#>
    + Engine_<ClusterID>_<Instance#>
    - Feeder_<ClusterID>_<Instance#>
        ErrorLogs<#>
        Performance
    - OMTEngine_<ClusterID>_<Instance#>
        Error Logs<#>
        Performance
    + OutcomeListener_<ClusterID>_<Instance#>
    + RampUpEngine_<Cluster#ID>_<Instance#>
    + Reports
  - OMT
    <Table_Alias>.bad
    <Table_Alias>.dat
    <Table_Alias>.log
```

About OMT table files

This section describes the files created under the OMT folder. There is one file (or one set of files) per each OMT outcome table. They are listed by the alias assigned to the table through the Outcome Management Tool (OMT).

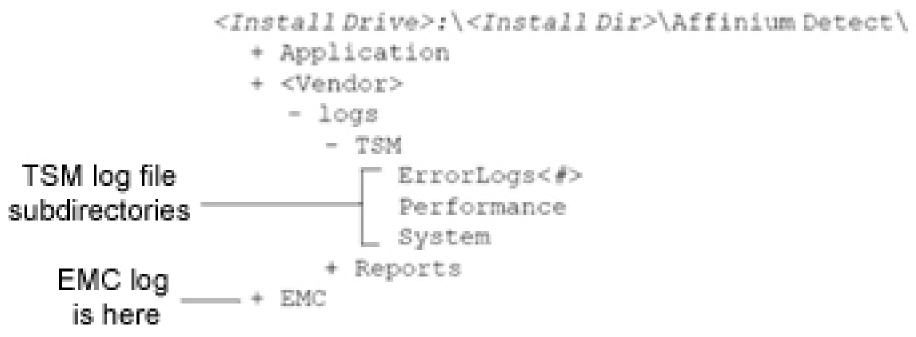
- `<Table_Alias>.dat`

  For Oracle and SQL Server, this file is the data extracted from the xml to be bulk imported to the database.
- `<Table_Alias>.log`

  Oracle-generated log file of what happened during bulk import.
- `<Table_Alias>.bad`

  Oracle-generated dump of record(s) that were not imported.

Sample directory structure on the web server unit



```
<Install Drive>:\<Install Dir>\Affinium Detect\
  + Application
  + <Vendor>
    - logs
      - TSM
          ErrorLogs<#>
          Performance
          System
      + Reports
  + EMC
```

Strategy for analyzing the log files

Different log files provide different information about possible failures. Knowing what was happening at the time of the failure can help you choose which log file to examine. For example, if the failure occurred before the system starts then check the EMC log directory.

In general, if the failure occurs during a run, begin by looking at the TSM/System log, which is used to monitor the behavior of the entire system. The TSM/System log may provide information that the UI cannot. For example, the UI may not be able to identify which server or cluster unit failed.

What to look for in the TSM log

The TSM log keeps track of the state of the whole system. Some best practices for using the TSM log are:

- Search for "feeder failed" or "listener failed" or any message preceded by five asterisks.
- If the feeder or the listener failed, go to its respective error log in order for more information about the error.
- If the engine failed, look through all of the engine error logs to find the problem. The error logs are located in the `*vendor*\logs\ Engine_*ClusterID_Instance#*\ErrorLogs1` directory under your Detect installation. Another way to approach the engine logs is by using the Detect Engine Trace Utility.

If the TSM log is missing or shows an incorrect timestamp

If you find that the TSM log is missing or it shows a timestamp that is from a previous run, then the system failed before it could create the TSM log. Look for the EMC log to diagnose this problem. Possible problems are:

- The EMC could not establish a connection to the database in order to load configuration data. Check the database configuration settings.
- The DCOM configuration from the EMC to the TSM is incorrect. Check the DCOM settings and permission settings.
- The arguments in the command line were missing or incorrect. Check the names of the feed files and the command line arguments.

About Detect log files

If the cluster goes down, a server in the cluster failed. The TSM log contains an error with the server ID. Go to the Configuration Utility and click Servers. The right pane contains a list of server IDs mapped to server names. Locate the name that corresponds with the server ID of the server that failed. In the left pane, click on the cluster (under the cluster node) and look for the server name to identify the cluster that failed. Go to the logs on that cluster to troubleshoot the problem.

Performance information in the TSM log

For obtaining performance information, refer to the TSM log.

There are periodic front and back engine performance reports of the running engine every 1,000,000 transactions. There is a front and back engine final report.

Failure Scenarios

This section describes the three failure scenarios and how to recover from them.

Failure due to wrong data

Failure due to wrong data typically occurs only during the development cycle. Examples are:

- Badly formatted specific piece of data, such as a data coming in at the middle of the file with wrong format.
- The data is entirely wrong or missing. For example, yesterday's files already ran but they are sill in the feed directory, or today one of the daily feed files is missing.

Failure due to faulty logic

Failure due to wrong logic typically occurs only when new triggers are being developed outside the production workspace. When this occurs, you need to determine type of failure. See "Recover Option 2."

Failure due to equipment failure

Equipment failures include failures in which the server stays alive but there is a network failure. For example:

- All servers in all clusters will shut down gracefully with current bookmarks of where they are in processing.
- Failures in which equipment catastrophically crashes such as a failed disk drive or power supply.
- True loss of data in memory that causes a corruption of state, even if the run is resumed at the nominal bookmarks.

See "Recover Option 1."

Recover Options

This section describes methods you can use to recover from the failures described in "Failure Scenarios".

Recover Option 1

1. Correct hardware problem (such as the downed network).
2. Start the run again in recover mode which causes each server in each cluster to pick up where it left off.

Recover Option 2

This recover option is the reason you should back up the state table nightly after every batch run. Backing up the state table enables you to restore it after a failure of this type.

1. Correct any possible hardware problem if it exists (such as server failure) or correct bad data in the feeds.
2. Restore state table from a backup.
3. Start the run again from the beginning.

Troubleshooting using the TSM log

This section describes what to search for in the TSM log.

Message indicating that the system initialized successfully

***** All system are up *****

These messages indicate a failure during run-time

<subsystem name> in failed state: <failed state> *****

Look for the error log under the OutcomeListener\errorlog folder

These messages could happen at any time (init, run, or shutdown)

***** Transaction Store Listener failed *****

Look for the error log under the TSM\errorlog folder

TSM log example

Detect is made up of a number of subsystems, which all must initialize and communicate with each other in order to have a fully functioning run. The TSM log monitors subsystem initialization and functioning of the overall system. The following is a TSM log that has been simplified in order to reveal its important aspects. (Comments are in bold.)

(The following are the states that the subsystems can be in. In the log, the states will be referred to by their number equivalents.)

***** State values are: *****

UNINITIALIZED is 1

INITIALIZING is 2

UP is 4

RUN_PROCESS is 6

RUN_END_SUCCESS is 007

RUN_END_SUCCESS_ERRORS is 11

RUN_FAILED is 8

STOPPING is 12

STOPPED is 9

STOP_FAILED is 13

INIT_FAILED is 15

CREATE_FAILED is 16

SHUTDOWN_FAILED is 14

SHUTTING_DOWN is 10

WAS_NOT_CREATED is 17

DOES_NOT_EXIST is 18

(Announcing various configuration settings that the system will use)

***** Use DB StopTimeLimit 300 *****

***** Use DB ShutdownTimeLimit 120 *****

(Announcing that a setting could not be found and that a default value will be used)

Cannot get defaultErrorTolerance in Dictionary; it is set to 10.

***** The default tolerance is 10 *****

***** The feeder tolerance is 1 *****

***** The engine tolerance is 1 *****

(Announcing that the feed delimiter is the pipe character)

TSM::InitFeeder, Got delimiter as |

(Announcing that the Listener and Feeder are initializing)

ListenerWrapper State is 2

Feeder Wrapper State is 2

(Announcing that the Listener is UP and the Feeder is still initializing)

Check if all system are in state UP

ListenerWrapper State is 4

Feeder Wrapper State is 2

(Announcing that the Listener and Feeder are both UP)

Wait for all system to be up

ListenerWrapper State is 4

Feeder Wrapper State is 4

(Announcing the state history size)

Init Visitor Size to: 11265

(Announcing that all subsystems have been initialized)

```
***** All system are up *****
```

(Announcing that the engines are starting up their various components. If there are multiple engines, there will be a lot of these messages)

```
Engine - Matcher Pipe Started

Engine - Front Engine Start

FrontEngine - Start Operation

Starting to listen to queues....

Engine - Back Engine Start

BackEngine - Started
```

(Announcing that 3 engines have been started and the application is going to start processing transactions)

```
About to start run. Engine count is 3
```

(Announcing that it found three feed files for the ‘a’ entity. The ‘a’ should match the type of entity specified in the run arguments)

```
STARTING FEEDS for files, dated: 20060620

STARTING FEED of Entity Type: a

found file: \\fcmddalcee1win\cee_inbound\elity.a.profile.20060620

found file: \\fcmddalcee1win\cee_inbound\elity.a.ca.20060620

found file: \\fcmddalcee1win\cee_inbound\elity.a.financial.20060620
```

(Announcing that the Listener and the Feeder are in the ‘RUN PROCESS’ (running) state)

```
ListenerWrapper State is 6

Feeder Wrapper State is 6
```

(Announcing the average ‘load’ time and the average ‘update’ time for reading and updating state history. You will see a ‘load’ and ‘update’ message for each engine.)

```
Load Average Time = 18.627211, Count = 1000, Max Buffer Length = 884

Update Average Time = 10.873696, Count = 1000, Max Buffer Length = 908
```

(Announcing a front engine report. It has seen 95,509 transactions and generated 40,000 simple events. It gives the average number of transactions per second. These will be issued periodically and there will be a report for every engine running)

```
Front Engine Report:
```

Transactions Count = 95509,

Event Counts = 40000,

Full-Pipe Count = 116,

Transactions per Second (Average) = 87,

Transactions per Second (Current) = 88

(Announcing a back engine report. It has processed 40,000 simple events. It is processing at the rate of 29 transactions per second. These reports will be issued periodically and there will be a report for every engine running)

Back Engine Report:

Events Counts = 40000,

State-Based Average Time = 20.578,

Empty Pipe Counts = 0,

Transactions per Second (Average) = 29,

Transactions per Second (Current) = 29

(Something has gone wrong. The Feeder has failed (8). With one subsystem failing, the system will now shut down all other subsystems and shut down the application)

Feeder Wrapper State is 8

***** feeder failed *****

(Final front engine report. There will be one report for every engine running.)

Final Front Engine Report:

Transactions Count = 168572,

Event Counts = 59107,

Filtered Transactions Count = 109465,

Exhausted-Pool Count = 0,

Full-Pipe Count = 88,

Empty-Queue Count = 12,

Transactions per Second (Average) = 90,

Front Engine Overall Run-Time = 1861:703 Seconds

(Summary report for the listener. Messages refers to a package of triggers that fired for one entity. Transactions refer to the total number of firings. The point

is that the application packages all firings for each entity into a single message and then the outcome listener unpacks the package and writes each trigger to the database.)

Listener stopped.

Processed 17781 transactions.

Received 13301 messages.

Error count: 0

Timing Information:

Location, #Trans, AvgTime(secs)

Full Run, 0, 0.000000

Single Trans, 13449, 0.138512

Getting Buffer, 17781, 0.000322

insertion, 17781, 0.020495

(Final back engine report. There will be one report for every engine running.)

Final Back Engine Report:

Events Counts = 56784,

State-Based Match Average Time = 17.347,

Dumped Transactions Count = 0,

Empty Pipe Counts = 0,

Transactions per Second (Average) = 30

(Announcing that the Listener is "shutting down" (12) and the Feeder has "failed" (8))

ListenerWrapper State is 12

Feeder Wrapper State is 8

***** All system has stopped in stop operation. *****

In shutdown, waiting for subsystems to shutdown

(Announcing that every subsystem is shut down and uninitialized.)

***** In final release, state is UNINITIALIZED *****

End of log messages report

About the Archive Service

Preferences for the Archive Service, including the number of days to retain system run information are set using the Configuration Utility.

Purpose of the Archive Service

Detect generates a number of logs during a batch run. These logs are scattered in multiple servers/clusters. The system overwrites these logs in any consecutive batch run unless you use the archive service. The archive service provides a means to collect and store these logs in an organized fashion per run. The archive service is part of the EMC subsystem and is executed at the end of each run.

Archive service capabilities

The archive service has three capabilities:

- Archive all logs from each cluster server per run
- Copy the batch run feed files
- Export outcome records from the outcome table stored in the database to a comma delimited file

Requirements for the Archive Service

This section describes requirements that must be met for the Archive Service to work.

Copy logs

In order for the archive service to work properly, every folder under your Detect installation in every cluster must be shared (with at least read only permission).

Copy feed files

In order for the archive service to work properly, every `\\<`*`cluster server`*`><`*`install dir`*`>\feed` folder must be shared (with at least read only permission).

The feed folder setting in the Configuration Utility should be set using the UNC format (`\\<`*`server name`*`>`) and not local notation (`C:\<feed path>`).

If only one cluster is defined and the user-specified feed folder in the Engine Manager, the archive service will use this location to search for the feed to copy. In this case the Archive service will work with the local notation as well (`C:\feed` path).

Extract otcome

The system supports writing outcome messages to more than one table. The system is installed with `<vendor>_<pub>_outcome table`.

You may choose to write to a second or other table (the format is `<vendor>_<workspace>_<my name>`).

The tool will extract information from any table listed in the `<vendor>__ActionDestinations` table.

The content of `<vendor>__ActionDestinations__ActionDestinations` is available when editing an Action component in the Destination drop down list.

Configuring the archive service

The archive service is configured using the Advanced Configuration settings in the Configuration Utility. (To see the settings, you must log into the Configuration Utility as a user with administration privileges.)

Archive service preference settings

For information about the archive service preference settings, see Appendix F, "Advanced Configuration Settings in the Configuration Utility," on page 159

About the Archive Extract Outcome setting

This section provides details about the Archive Extra Outcome setting.

All

The All setting tells the system to export the content of all configured outcome tables (<outcome user name>.<vendor>_<workspace>_<destination name>) and export the counts of firing for each component id.

The file names are:

```
<Vendor>_<workspace>_<destination name>.txt
<Vendor>_<workspace>_<destination name>_Count.txt
```

The format is comma delimited. For example:

```
ENTITYID, ENTITYTYPE, RULEID, MESSAGE, FIRINGTIME, PROCESSED
IM000010159077000000000000000,97,345,<OUTPUT><TEXT>Done</TEXT>
</OUTPUT>,4/21/2005,1
IM000010335576000000000000000,97,345,<OUTPUT><TEXT>Done</TEXT>
</OUTPUT>,4/11/2005,1
IM000010336157000000000000000,97,345,<OUTPUT><TEXT>Done</TEXT>
</OUTPUT>,3/4/2005,1
```

and

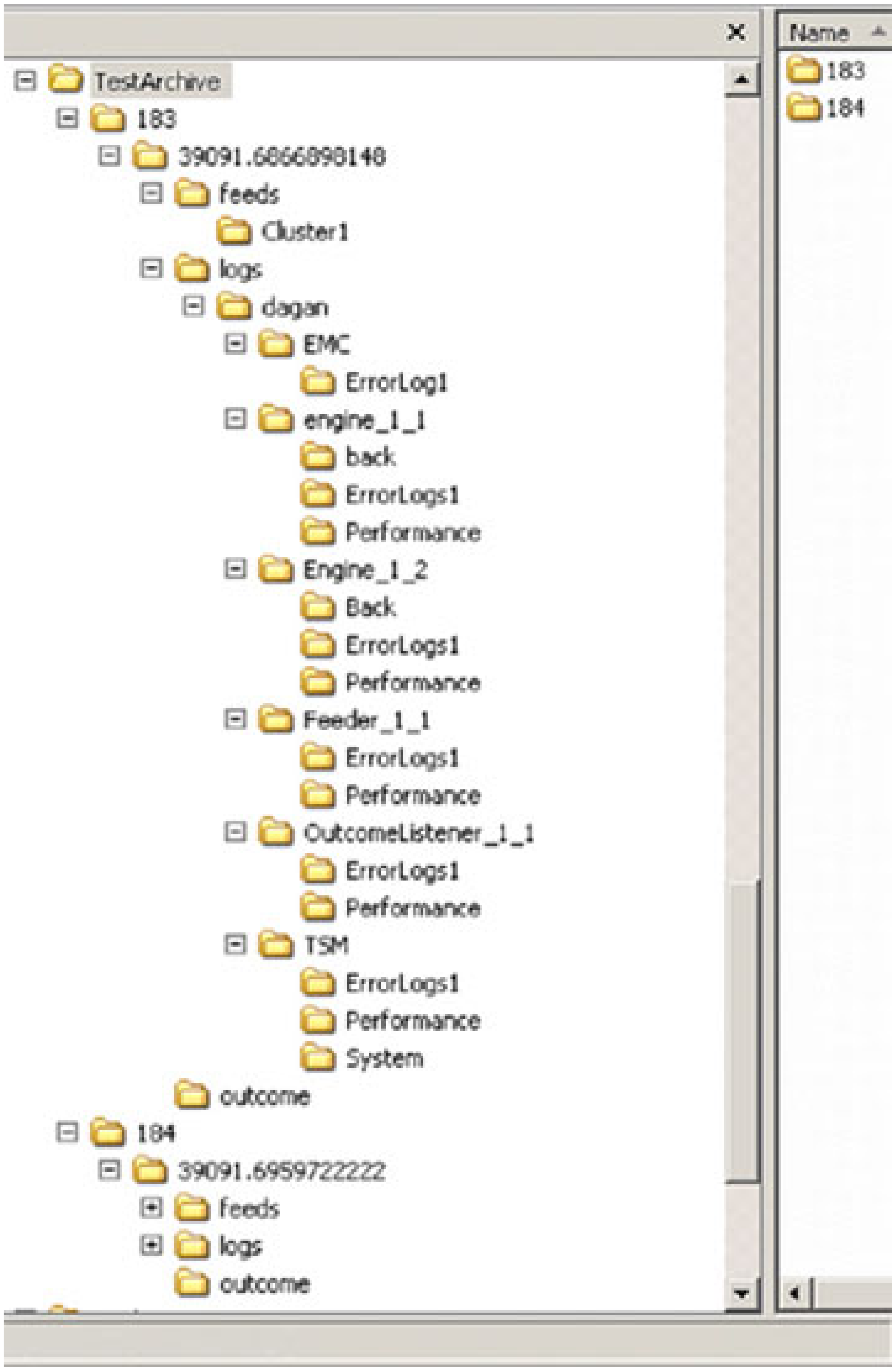
```
RULEID, COUNT(*)
 345,1058
```

Counts

If Counts is selected only the `<Vendor>_<workspace>_<destination name>_Count.txt` file is generated.

Example for directory structure

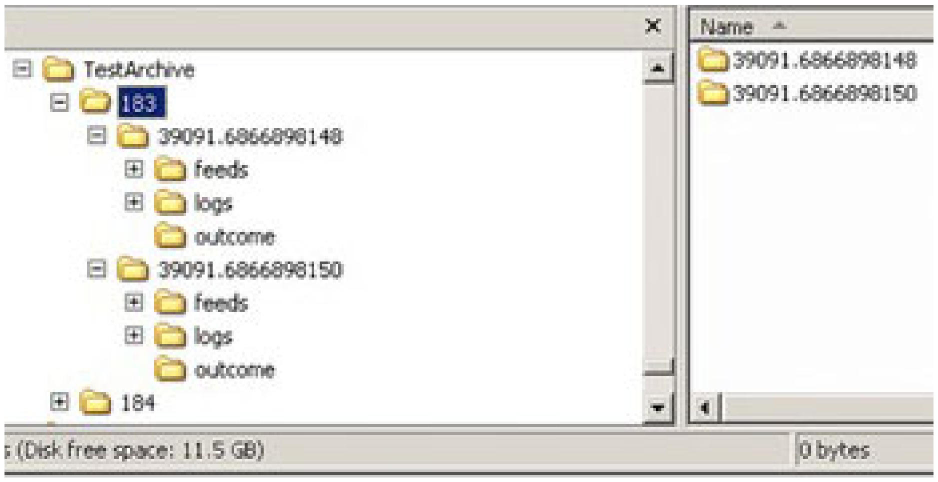
This section shows an example for the structure of the archive directory.



Recoverability

In a recovery scenario the runid folder will not change but a new folder named with the new recovered run timestamp will be created to holds log from the second run (and any consecutive attempt to recover)

Example for the recovery case:



Appendix E. Database Schemas, Users, and Logins

This section describes basic concepts about database schemas, users, and (for SQL Server) logins. The Detect database setup instructions often mention these concepts.

About schema and database user names

The names given to the Rule, History and Outcome schema user names during IBM Unica Detect installation are used to define the database user and schema, as well as the login if the database is SQL Server.

About schemas

A schema is a logical container for the database objects, such as tables, views, and triggers.

About database users

A valid user account within a database is required to access that database. User accounts are specific to a database. All permissions and ownership of objects in the database are controlled by the user account.

About database user and schema names in Oracle

In Oracle, database users and schemas have a one-to-one relation. When creating a user, a schema for that user is also implicitly created. The schema name is the same as the user name, and can be used to unambiguously refer to objects owned by the user.

About database user and schema names in SQL Server

In SQL Server, database users and schemas were conceptually the same object in previous versions of SQL Server. However, beginning in SQL Server 2005, users and schemas are separate. A schema can be owned by any user, and schema ownership is transferable.

About logins

A login is an SQL Server concept. In SQL Server, a login is used for user authentication, while a database user account is used for database access and permissions validation. During a new connection request, SQL Server verifies the login name supplied, to make sure that the login is authorized to access SQL Server.

A login is defined at SQL server level. Any database user has to associate with a login. A login can have associated users in different databases, but only one user per database.

About login security modes

There are two security modes for a login in SQL Server 2005: Windows Authentication and SQL Server Authentication. Detect uses SQL Server Authentication model.

About the Detect database schemas

To serve the different needs different types of data, the Detect database is made up of three named schemas:

- Rule — includes component definitions, machine states, system configuration information, and logging information as well as all the global information used by the system.
- History — stores visitor history and state information.
- Outcome — contains the outcomes from components triggered during a run.

Database user and schema name convention

Because Detect supports both Oracle and SQL Server databases, for the sake of simplicity and maintainability, Detectt uses the same schema and user naming convention as Oracle. The schema name is the same as the user name and can be used to unambiguously refer to objects owned by the user.

For Detect with SQL Server, for simplicity, the default login name is the same as the database user's name.

About Detect database passwords

In Oracle, a password must be specified for each database user, while in SQL Server, a password must be specified for each login.

For simplicity, the Detect installer requires you to specify one password, called "Database Password". This password is used for all three database users (or logins for SQL Server). This password is also used to create three connection strings for three users in `%Vendor%__Connections` table.

To change the password for a schema user

Use these steps when you want to change the password for a schema user.

1. Use Configuration Utility to change the password for the user in the %Vendor% connections table.
2. For Oracle, go to Oracle Enterprise Manager Console and change the password for the user. For SQL Server, go to SQL Server Management Studio and change the password for login of the user.
3. If the user is the Rule user, also update the password in the registry. You can use PasswordTool.exe (located in the `Application\bin` directory under your Detect installation). Type the new password, click **Encrypt**, and then click **Update Registry**.

Appendix F. Advanced Configuration Settings in the Configuration Utility

The following tables describe the configuration properties on the Advanced Configuration tab of the Configuration Utility in Detect.

Note: Items marked with an asterisk (*) do not revert to default settings if you click the **Restore Defaults** button.

Archive Service

These settings control the level of archiving.

Setting	Description	Default
Archive Feed Files	The archive process copies the feed files to the archive folder if the value is set to **YES**. The system copies the feed files from the feed folder (per cluster) to the folder: `\\`*`Archive Server`*`\`*`Archive Share`*`\`*`runID`*`\`*`timestamp`*`\feeds\`*`Cluster Name`*`\`	NO
Archive Run Logs	If this value is set to **YES**, the archive service copies all run logs to the archive folder, organized by server name. The system copies all logs to the folder:`\\`*`Archive Server`*`\`*`Archive Share`*`\`*`runID`*`\`*`timestamp`*`\logs`	NO
Archive Server	The name of the server on which to store the archive logs, feeds, and outcome.	server name
Archive Shared Folder	The shared root folder on the archive server in which to archive the files. The folder must be a shared folder, with read and write permissions.	C$ *

Setting	Description	Default
Days for Archived Logs	The amount of time, in days, to keep run history, logs, and archive files. Data older than the number of days set is automatically purged from the following tables: • G_Run • G_Runsystemstats tables • G_Enginebookmarks • G_Errors • G_Subsystemstatus • G_Subsystemperf • G_Logs • G_Runstatus • G_Rulefiringcounts **Note:** The system purges archived files based on this setting. Archiving and purging date calculations are done based on the current machine's date, not on feed dates.	365
Outcome Extraction	Whether or not to extract outcome from the outcome table. The drop-down list has these choices: • **All**—Archive the entire outcome table, including every record and all the fields • **Counts**—Archive the firing count, grouped by Action ruleID • **None**—Do not archive any data in the outcome table.	None

Database

View or change the database configurations.

Setting	Description	Default
Database Type	The ID of the database type from the G_Databases table, which indicates 1 for Oracle and 2 for SQL Server.	(Controlled by installation scripts) *
Outcome Database User Name	Used by SQL server to connect to the outcome table. This is the one schema that can reside outside the Detect database.	(Controlled by installation scripts) *
Outcome Database Vendor	The ID of the outcome database type from the G_Databases table, which indicates 1 for Oracle and 2 for SQL Server.	(Controlled by installation scripts) *

Error Tolerance

These settings define the maximum number of errors that can occur for a run to be considered successful.

Setting	Description	Default
Engine Error Tolerance	The maximum number of errors that the engine can encounter during processing before the system stops and shuts down.	1
Feeder Error Tolerance	The maximum number of errors that the feeder can encounter during processing before the system stops and shuts down.	1
Listener Error Tolerance	The maximum number of errors that the listener can receive before the process stops and shuts down.	1

Locale

View or set the locale.

Setting	Description	Default
Locale Identifier	Sets the date format for the following. • Reports • Inactivity target date of Engine run • Constants in Simple or Qualifier editors • Outcome messages • Outcome Management Tool parsing of dates from outcome tables • Timestamps in Rampup feeds Use the drop-down list to see and select the options.	ISO 8601 Date Format *

Logging or Reporting

Setting	Description	Default
AlwaysOn Log Listener Port Number	This port number is used by the log server and log clients to communicate log messages when the engine runs in always-on mode. It must be an available port on the server that has the TSM.	4935
AlwaysOn Performance Listener Port Number	This port number is used by the log server and log clients to communicate log messages when the engine runs in always-on mode. It must be an available port on the server that has the TSM.	4945
Error Logs Persistent Method	This setting controls where to write the error logs. May be set to Database or File. By default, the error logs are written to the database to be viewable within the engine manager.	Database
Log Listener Port Number	This port number is used by the log server and log clients to communicate log messages when the engine runs in batch mode. It must be an available port on the server that has the TSM.	4915

Setting	Description	Default
Performance Listener Port Number	This port number is used by the log server and log clients to communicate log messages when the engine runs in batch mode. It must be an available port on the server that has the TSM.	4925
Save Firing Counts to the Database	This is an optional setting that turns on or off the gathering of component firing information to be used in reports	On
System Log Persistent Method	This setting controls where to write the system run time log information. May be set to Database or File.	File
TSM Log Rotation Interval	The number of messages the TSM will log before rotating the file. A value of zero results in no rotation. If you run in Always-on mode, you should set this value to 1000 or greater..	File

Notifications

These settings define the names of files Detect attempts to run after various processes complete. You can change the names of the specified file here, but if you do, you must supply a file of that name. These settings are typically used when a separate system is set up to automatically run other batch jobs or reports.

Setting	Description	Default
Alert File Name	This file is executed by the EMC N seconds before the end of any always-on run, if the **Alert Before Processing Completion** option in the **System Options** section is set to a value greater than 0.	alert.bat
AlwaysOn Initialization Failed	This file is executed by the EMC if an attempt to initialize an always-on run mode fails.	initFailed AO.bat
AlwaysOn Initialization Success	This file is executed by the EMC if an attempt to initialize an always-on run mode succeeds.	initSuccess AO.bat
AlwaysOn Shutdown Failed	This file is executed by the EMC if an attempt to shut down an always-on run mode fails.	shutdown Failed AO.bat
AlwaysOn Shutdown Success	This file is executed by the EMC if an attempt to shut down an always-on run mode succeeds.	shutdown Success AO.bat
AlwaysOn Transaction Failed	This file is executed by the EMC if a run in always-on mode fails.	runFailed AO.bat
AlwaysOn Transaction Success	This file is executed by the EMC if a run in always-on mode succeeds.	runSuccess AO.bat
AlwaysOn Transaction Success with Errors	This file is executed by the EMC if a run in always-on mode succeeds with errors. The errors in this case would not be enough to shut down the system. (See related settings in this table, listed under Error Tolerances.)	runSuccess WithErrors AO.bat

Setting	Description	Default
OMT Failed	This file is executed by the EMC if the OMT Populate Outcome operation fails.	omtError.bat
OMT Success	This file is executed by the EMC when the OMT Populate Outcome operation finishes successfully. You can use this file to automate down stream processing of the application outcome.	omtDone.bat
PreProcessor Failed	This file is executed by the Preprocessor if it fails.	PreProcessor Error.bat
PreProcessor Success	This file is executed by the Preprocessor when it runs successfully. You can use this file to automate transaction processing.	PreProcessor Done.bat
Ramp Up Failed	This file is executed by the EMC if the Ramp Up process fails.	rampup Error.bat
Ramp Up Success	This file is executed by the EMC when the Ramp Up process finishes successfully.	rampup Done.bat
Transaction Failed	This file is executed by the EMC if a run in batch mode fails.	ftpError.bat
Transaction Success	This file is executed by the EMC when the system finishes a run in batch mode successfully. You can use this file to automate down stream processing of the application outcome.	ftpPut.bat
Transaction Success with Errors	This file is executed by the EMC if the system finishes a run in batch mode successfully with errors. The errors in this case would not be enough to shut down the system. (See related settings in this table, listed under Error Tolerances.)	ftpPutWith Errors.bat

OMT Options

Setting	Description	Default
Float Field Precision	Sets the numeric precision and scale for floating numbers used by the Outcome Management Tool. Applies to systems using the Oracle database only.	(15,5)
Integer Field Precision	Sets the numeric precision and scale for integer numbers used by the Outcome Management Tool. Applies to systems using the Oracle database only.	(10,0)

Processing Options

Setting	Description	Default
Container Aging	How to calculate the cutoff time for container aging. O: Exclusive, 1: Inclusive	Exclusive *

Setting	Description	Default
Feeder Data Validation	If true, the feeder validates the data being processed against the associated data type defined within the data source. If the data type does not match the data, then the feeder generates an error and quit.	False
Feeder Throttle	This setting turns on and off the Feeder Throttle. The options are On and Off. **Note:** When Feeder Throttle is set to On, ensure that the Feeder Pause Interval parameter is set to 0 (zero, for off). The Feeder Pause Interval parameter is a less effective mechanism for controlling the number of messages in the queue. The Feeder Throttle is a monitoring capability for the Feeder. It prevents messages from backing up in the engine queues and exhausting the MSMQ service resources. The Feeder Throttle controls the number of messages on the MSMQ Service during a batch run. If there are more than a certain number of messages, the Feeder sleeps. If the engines are fast enough and keep up with the feeder, the Feeder Throttle ensures that the Feeder never sleeps (slowdown).	On
Float Operation Accuracy	This setting is used by the back engine when dealing with floating numbers and significant digits. It is used to determine the tolerance level by the engine when dealing with rounding issues.	12 *
Inactivity Feed Path	Inactivity Feed Path holds the location of a shared folder where Detect can write a file used during inactivity processing. With this option enabled, a single query gathers inactivity data, rather than queries from every Feeder in a cluster. If you set a value for this option, you can also choose to have them automatically deleted by setting the Delete Inactivity Files option in the Archive service section. Set the value to the path to a shared folder. For example: `\\servername\sharename` If you do not set the value of the the Inactivity Feed Path configuration option, multiple feeders can perform inactivity queries. However, synchronization across clusters still occurs; no engine processes a transaction until all Feeder queries for inactivity events are finished processing.	

Setting	Description	Default
Inactivity Mode	This setting controls how the engine processes inactivity events. **Disabled Firing**–All arming FLI events are processed and all firing events are suppressed by the engine. **CRM**–Process all transactions prior to processing inactivity events. **RealTime1**–Process inactivity events between transactions. It tells the system that before processing the transaction, check to see if any inactivity events are due to fire before the transaction date. If there are multiple transactions, check for inactivity events before each transaction. In this mode, the feeder checks for inactivity events prior to processing transactions. If processing files for multiple days at once, it checks for inactivity events between processing each day's transaction feeds **RealTime2**–The engine behavior is the same as for the RealTime1 option. However, with this option the feeder does one final query based on the inactivity end date. The inactivity end date is a parameter set in the command line or in the Engine Manager. In this mode the following occur: • The feeder checks for inactivity events prior to processing transactions. If processing files for multiple days at once, it checks for inactivity events between processing each days transaction feeds. • The system does one final inactivity processing pass, based on the target date (inactivity end date). • This mode (and the Process Between Transactions mode) are the most accurate and thorough. **Off**–No inactivity events are processed.	RealTime2 *

Setting	Description	Default
Max Messages To Send Without Monitoring	This setting determines the maximum number of messages to send before monitoring. This number correlates to the X in the algorithm. So, for example, when the value is set to 60,000, the total number of messages in all the engine queues should not be more than 90,000. **Note:** If this parameter is set to a very high value, the feeder never checks the queue. Setting the parameter to a high value is one way to bypass the mechanism on a daily run. If this parameter is set too low, the checks are too frequent and may slow the feeder to the point that the engine is waiting, thus starving the engine.	60,000
Sleep Time Maxed	This setting controls the Feeder's sleep time when the message count is equal to 1.5 times the setting for **Max Messages To Send Without Monitoring**. (This sleep time is rarely invoked.)	500 milliseconds
Sleep Time Not Maxed	This setting controls the Feeder's sleep time when the message count is below 1.5 times the setting for **Max Messages To Send Without Monitoring**. If this value is set too high there is a chance of starving the engine.	30 milliseconds

System Options

Setting	Description	Default
Alert Before Processing Completion	Amount of time, in seconds, before the engine completes an always-on run, when Detect attempts to execute the batch file specified by the **Alert File Name** option. If this value is set to `0`, Detect does not attempt to execute the batch file.	300
Bookmark Frequency	The engine writes a bookmark record to the G_Enginebookmark table after N number of records	1000
Command Timeout	Sets the command timeout when querying the database in critical areas.	300

Setting	Description	Default
Default input file encoding	Replaces the former Default Feed Encoding property used in previous releases. Sets the encoding for the following. • State snapshot reports • The input file for the Presentation Layer Manager • Ramp-up feed files • `DebugIds.txt` file for the engine • `SuppressedRules.txt` for the engine Use the drop-down list to see and select the options. All log and other files generated by Detect are saved in UTF 8 encoding, regardless of any configuration settings.	
Delete Inactivity Files	Specifies whether the feeder deletes old inactivity files. Deletion takes place if the value is set to YES.	NO
Estimate Processing Time	Specifies whether Detect calculates and monitors the time remaining until the run completes.	YES
Excessive Transaction Threshold	The Feeder logs users that have transaction counts that exceed this threshold.	2000
Feed Delimiter	The field delimiter used by the feeder to parse the data files	|
Feeder Pause Interval	This setting is used to slow down the feeder from processing too quickly. A value of 0 means the feeder does not sleep.	0 *
History Buffer Size	Size of the entity state buffer in bytes. Maximum value is 10000. This setting has a direct effect on performance.The larger the buffer, the more it affects performance.	11263 *
IIS Instance determined by	You need to change this setting only when you set up multiple environments and you want to use a different instance of IIS for each environment. If that is the case, select **Environment.** If you set up multiple environments and want to use one instance of IIS for all of the environments, leave the default value of **System**.	System
Inactivity Query Timeout	Sets the number of seconds that Detect waits before terminating an inactivity query.	300
Progress Recalculation Interval	The interval, in seconds, between calculations that Detect performs to determine the progress of a run and the remaining time until the run completes.	120

Setting	Description	Default
Simultaneous runs	Enables simultaneous batch runs. The following options are available. • **Off**–Only one run is allowed regardless of the number of environments. • **Contention Support**– Multiple runs are allowed, including multiple runs on the same workspace. The system protects itself from potential deadlocks in the state table. • **No Contention Support**–Multiple runs are allowed, but not on the same workspace.	Off
Time to Initialize	The maximum time allowed, in seconds, for the system to initialize.	300
Time to Shutdown	The maximum time, in seconds, allowed to shut down the system.	300
Time to Stop	The maximum time, in seconds, allowed to stop the system.	300
Transaction Packet Size	The feeder packages messages after N number of messages for a user. If the transactions for a user exceed the set value, then the transactions are split into separate packages to be sent to the engine for processing. You probably never need to adjust this setting.	2000
TSM Timeout	The time, in seconds, that the EMC waits for a response from the TSM. If this time period has expired then the EMC shuts down the system.	300

Appendix G. Glossary for IBM Unica Detect

A	
Action (component)	A component that communicates defined pieces of data (customer number, time, reason to contact) that are written to a database when a specified event occurs. Represents a recommendation to contact a customer due to a pattern of behavior.
Artificial Transaction (ATX)	A transaction created during run-time, based on an optional parameter of the engine process. An Artificial Transaction can be specified as a significant event in a Pattern component or in a Container Manipulator component. Also available in Action.
B	
Backward Looking Inactivity (component)	A component to detect 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 Looking Inactivity component fires if the event being looked for did not occur.
C	
Component	The title given to each of the different types of building blocks that process events and data within Detect: Simple Event, Qualifier, Math, Pattern, Pattern Expression, Backward Inactivity, Forward Inactivity, Join Function, Select Function, Container, Container Manipulator, Action components.
Container (component)	A bucket of information of the same type. The type (structure of a row of information) is determined by a previously defined Type Descriptor. The data stored in the Container may come from transactions, lookup tables, math expressions, select functions, containers, and / or may be the result of the firing of a Container Manipulator.
Container Manipulator (component)	A component designed to manipulate data in containers. Container Manipulators may insert and delete data in containers.
D	
Data Source	A source of input data for the system, such as profile file, transaction file, or lookup table. There can be only one profile source per entity, whether from a profile file or lookup table. There may be any number of non-profile lookup tables and there may be any number of transaction files. However, because there must be at least one transaction data source per entity.
Data Source Field	Each field within a data source that describes the name and the data type.
E	
Entity	An individual element that the system tracks and saves historical information for, such as Account, Customer, Household, etc.
Event	The result of the data being processed causing a true condition of a component. Detect recognizes the trigger event and evaluates whether its occurrence is significant to any other components within a trigger system.
F	
Feed File	A single-delimiter flat file containing transaction information for each entity for a specific time duration (usually a day).

Forward Looking Inactivity (component)	A component to detect 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 Looking Inactivity component fires when the time period expires without the occurrence of the event it for which it was waiting. The tracked event is optional.
J	
Join Function	This component that enables the user to join two record-set objects. The Join is based on a field common to both data sources. The Join Function returns a record-set object whose type has been pre-defined using the Type Descriptor Editor.
Lookup Table	A user-defined set of fields that describe a table of information. It is defined as a database data source that is not associated with an entity.
M	
Math (component)	A component defined as a mathematical formula that calculates a numeric result based on numeric data from one or more data sources that can be used in Simple Event components, Qualifier components, Container Manipulator components, and other Math components.
P	
Pattern (component)	Defines a group of events occurring within a defined time frame. For example, if three Printer Purchase" events occur within one week (for a given customer) then fire Multiple Purchases" pattern.
Patterns	The term given to the larger category of component types that monitor the occurrence of events over a period of time, thereby maintaining state". These include; Pattern, Backward Inactivity, Forward Inactivity.
Profile	A set of mostly static attributes used to characterize an entity while processing transactions. May be provided by the profile file or by an external profile table.
Profile File	A single delimiter flat file containing (relatively static) profile information for an entity.
Q	
Qualifier (component)	A component that defines a set of customer characteristics that reveal a quality about the customer as opposed to recognizing the behavior of that customer. For example, if the customer's gender = male and the customer's city of residence = New York City, then the "Men from NYC" Qualifier fires. Qualifier components act as filters, refining the firing of Simple Event components, Action components, and Container Manipulators to those customers who posses the defined quality. Qualifiers may be constructed using any data source in the system.
S	
Select Function	This component operates on a single record-set object. It provides a means of reducing the data in a table by using a select plus a where clause.
Simple Event (component)	A defined set of business-relevant conditions that recognize certain attributes in a particular transaction. E.g. if SKU # =100 then fire the "Printer Purchase" event.
State History	A record of all event activity and data storage such as a component firing or event in the past.
T	
Transaction	A single line item captured by an operational system. Raw data" supplied by IT representing an interaction.

Transaction Set	The set of transactions for an individual entity for a time duration (usually a day), plus their profile entry.
Trigger	Trigger has two meanings. • The act that occurs, to a component within Detect, when the data being processed causes a true condition of that component. The component is said to trigger, or to trigger an event, or to fire. • A hierarchically organized relationship of components, used as building blocks, whose purpose it is to recognize specific customer behavior or attributes. Also called Trigger System. Every Trigger System starts with at least one Simple Event component. A complete Trigger System culminates with an Action component.
Type Descriptor	A definition of a record-set type. The type holds a vector of simple types with accompanying names for each of the positions.
V	
Vendor	A Detect customer.

Notices

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