Version 10 Release 0
February 28, 2017

IBM Opportunity Detect Administrator's Guide

Note
Before using this information and the product it supports, read the information in "Notices" on page 125.

This edition applies to version 10, release 0, modification 0 of IBM Opportunity Detect (product number 5725-D16) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Chapter 1. About IBM Opportunity Detect

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

You use Opportunity Detect components to build trigger systems in workspaces. When you run a workspace, its trigger systems apply your business logic to streams of data from your transaction and profile data feeds.

Opportunity Detect uses IBM InfoSphere® Streams technology for enhanced performance.

Streams is an advanced analytic platform that allows Opportunity Detect to quickly ingest, analyze and correlate information as it arrives from batch and real time sources. The solution can handle very high data throughput rates.

Integration with IBM Campaign

You can integrate Opportunity Detect with IBM Campaign.

You can use the Expanded Outcome data source connector to store the Outcome data in database tables that Campaign can use. See the section on Expanded Outcome tables in the *IBM Opportunity Detect User's Guide* for details about this integration.

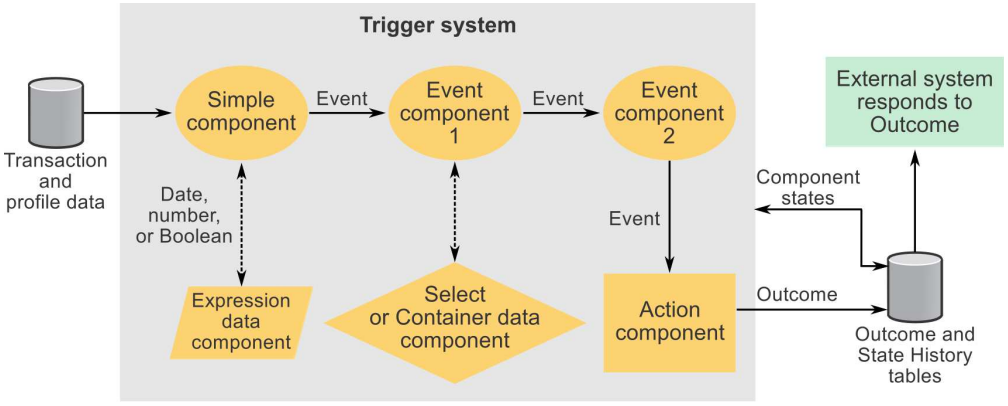
Note: Another product, IBM Interact Advanced Patterns, provides integration with IBM Interact. With this integration, you can apply Pattern component logic to data sent from Interact in real time. See the *IBM Interact and IBM Interact Advanced Patterns Integration Guide* for details.

Trigger system basics

A trigger system is comprised of configurable components. When a trigger system runs, it produces State History and Outcome data.

State History data is the saved results of processing some components; the system uses this information in subsequent runs. Outcome data can be used by external systems to respond to customer activity.

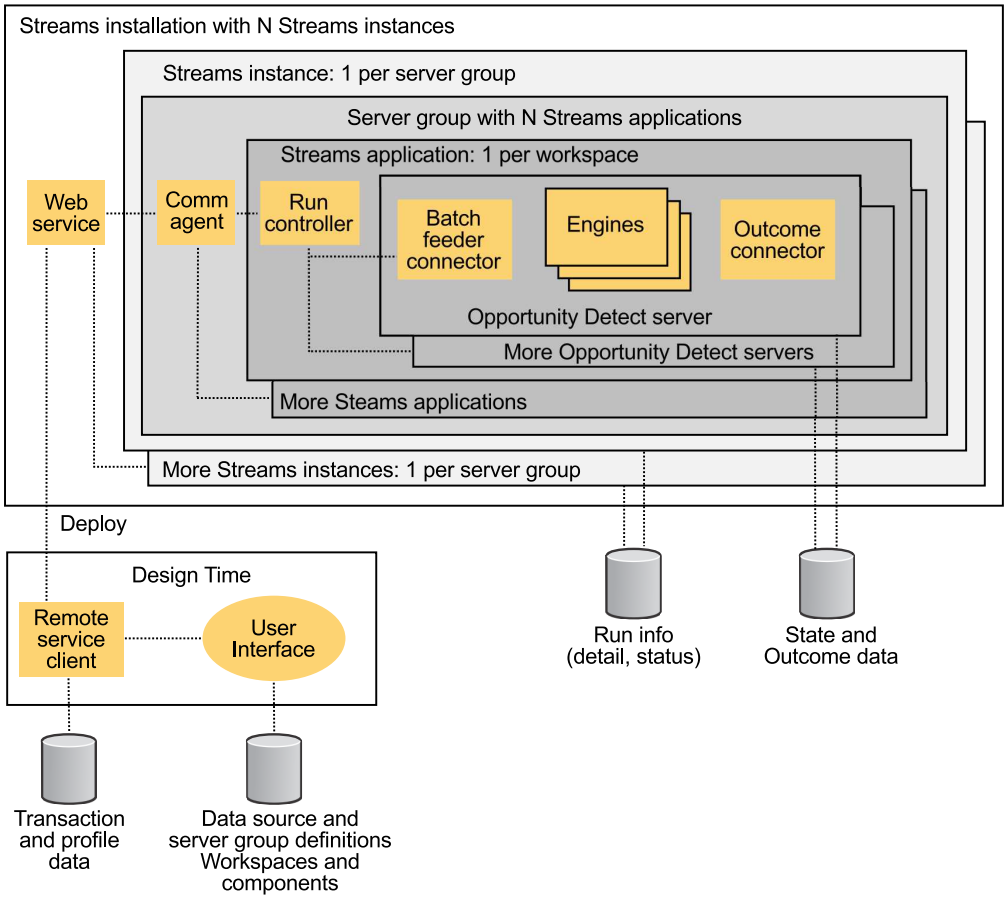
The following diagram illustrates a basic trigger system.



IBM Opportunity Detect architecture

IBM Opportunity Detect uses IBM InfoSphere Streams, a high-performance computing platform. Multiple instances of the Opportunity Detect engine can run in servers in a Streams instance. You can define multiple server groups and specify the data they can access. You can also specify which server group processes your Opportunity Detect workspace.

The following diagram provides a high-level overview of Opportunity Detect architecture for batch mode.



Locale in Opportunity Detect

Opportunity Detect supports several locales for transaction and profile data.

When you create a File type of data source connector, you can select a locale for the data in these files.

Note that full support for any character set in Opportunity Detect also depends on the configuration of the following.

- The Opportunity Detect system table database.
- The client machines and browsers used to access Opportunity Detect.

Enable cookies

Enable cookies in your browser to take advantage of all of the features provided in the Opportunity Detect user interface.

Chapter 2. Opportunity Detect roles and permissions

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

You manage user application access by assigning the desired roles and permissions to individual users, or by assigning users to groups that have the desired roles and permissions. You can use the default roles, or create custom roles with the permissions that you specify. You can not create custom permissions, only custom roles.

You manage roles and permissions from the Users, User Groups, and User Roles & Permissions pages. All of these pages are available under the **Settings** menu.

Tip: For help when you work on these pages, click **Help > Help for this page**, or see the *IBM Marketing Platform Administrator's Guide*.

Permissions for Opportunity Detect

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

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

Table 1. Permissions in Opportunity Detect

Permission	Description
View only	Can access all of the user interface, in view-only mode.
Design triggers	• Can create workspaces and design trigger systems. • Can create, modify, and delete all trigger related resources. • Can access Workspace, Component, Audience Level, Data Source, and Named Value List pages. • Can not access the Server Groups page or the Deployment tab of a workspace. • Can not set off a batch run. • Can not administer objects that the web service creates when Opportunity Detect is integrated with Interact.
Run for testing	• Deploy deployment configurations and run batch deployment configurations on server groups not designated for production. • Can access Server Group page and the Deployment tab of a workspace, but can not designate a server group for production. • Can not deploy deployment configurations or run deployment configurations that use a production server group.
Run for production	• Deploy deployment configurations and run batch deployment configurations on any server group. • Perform all actions on the Server Group page and the Deployment and Batch Run tabs of a workspace, including designating a server group for production.

Table 1. Permissions in Opportunity Detect (continued)

Permission	Description
Administer real time	Manage objects that the web service creates when Opportunity Detect is integrated with Interact to enable real time mode. Allows the following. • Delete workspaces and components created by the web service. • Start and stop real time deployment configurations and update their log level. The user with this permission alone can not start runs for real time deployment configurations. No one, even with this permission, can do any of the following. • Delete and update audience levels, data sources, named value lists, server groups, or deployment configurations created by the web service. • Create and deploy deployment configurations created by the web service.

Related concepts:

"Built-in roles in Opportunity Detect"

Built-in roles in Opportunity Detect

Four built-in roles are included with Opportunity Detect.

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

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

Table 2. Built-in roles in Opportunity Detect

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

Related reference:

"Permissions for Opportunity Detect" on page 5

Chapter 3. Real time processing in Opportunity Detect

Opportunity Detect offers three data source connectors for processing transactions as soon as they are received.

The type of data source connectors that you use for your workspace determines whether it is processed in batch or real time mode. For real time operation, use a Real time file, Queue, or Web service type of data source connector for transaction and Outcome data.

Web Service connector

The Web Service connector is a good choice when moderate volumes of data are expected. In tests, this connector has been shown to handle up to 3,500 transactions per second.

To use the Web Service data source connector, your organization must develop code to receive and send the transaction and Outcome data.

Queue connector

A Queue connector is a good choice when a high volume of transactions must be processed.

Queues handle spikes in demand more efficiently than the Web Service does. Also, unlike the Web Service, queues retain messages in the event of a network or machine failure, or if you deploy a new version of a configuration deployment.

To use the queue data source connector, your organization must develop code to send the transaction data, and you must install and maintain a queue server.

Real time file connector

The Real time file connector handles fixed width ASCII files, processing transactions as soon as a file is placed in a designated folder. After processing, the file is placed in a different designated folder.

If you have more than one deployment using the Real time file connector, ensure that a different folder is designated for the transaction files for each deployment, to prevent an condition where the file is moved before one of the deployments is finished with it.

Opportunity Detect does not support live updates to files in the input directory used with the real time file connector. Also, you should not use the same input directory for multiple workspaces, as this could lead to undesired behavior.

When you configure the file connector, you can eliminate duplicate transactions by applying a Bloom filter to fields that you specify. You can also set the period of time for which duplicates are ignored.

The Bloom filter operator detects duplicate transactions in a memory-efficient way. Use the Bloom filter in scenarios that require duplicate detection for large numbers of unique transactions over a period of time. False positives are an occasional side effect of compression and occur when a transaction is marked as a duplicate even though it is unique. You can configure this operator with the number of expected unique data entries and the probability of false positives. If you have a

high number of expected unique data entries and a low probability of false positives, you need a higher amount of required memory.

This connector is particularly useful for Call Data Records (CDRs) used by the telco industry. The CDR must first be transformed from binary to ASCII format.

Data source connectors are mapped to data sources in two places. The default mappings are set when a server group is configured. However, these mappings are commonly changed when a deployment is configured.

Related concepts:

"About data source connectors" on page 28

Chapter 7, "Queue connectors for input and output in Opportunity Detect," on page 57

Chapter 8, "Web Service connectors for input and output in Opportunity Detect," on page 69

Chapter 4. Data source setup in Opportunity Detect

To set up data sources for IBM Opportunity Detect, you configure them in the user interface, including setting up connections to the databases that hold run detail, State History, and Outcome data.

To develop data feeds, you must identify the operational systems, databases, and tables that hold the feed data you want to use. If you plan to use flat files, you must extract this data into flat files in the required format, with the required file names.

The following table describes the input and output data you set up in Opportunity Detect.

Note: All of your database tables must be of the same type. If you use DB2 for your system tables, Oracle is not supported for profile or lookup tables. If you use Oracle for your system tables, DB2 is not supported for profile or lookup tables.

Table 3. Types and formats of data used in Opportunity Detect

Type	Description	Allowed formats
Input data Your profile and transaction files must be accessible from the machine on which the Opportunity Detect run time component is installed.		
Transaction data	Customer activities, obtained from your organization's operational system. At least one transaction data source is required to run a workspace.	• Flat file • A message from a queue server • A message from the IBM Opportunity Detect web service
Profile data	Customer attributes. Examples of profile data are age, home address, and telephone number. Optional.	• Flat file • Database tables If a queue or the Web Service data source connector is used for transaction data, the associated profile data must be in a database table.
Lookup data	An auxiliary table that holds static data, and is used to look up values.	• Database tables
Output data (system tables)		
State History	Data needed for component processing. You can share State History tables with multiple workspaces for development and test purposes, and set up one or more State History tables for exclusive use by the production environment.	• Database tables
Outcomes	Data produced by trigger systems, for use by external systems. You can share Outcome tables with multiple workspaces for development and test purposes, and set up one or more State History tables for exclusive use by the production environment.	• Database tables • A message from a queue server • A message from the IBM Opportunity Detect web service

Table 3. Types and formats of data used in Opportunity Detect (continued)

Type	Description	Allowed formats
Run detail	Data produced by the engine, containing information about workspace runs. The table is created in the run time database automatically during installation. When you set up data sources, you define a database connection for the run time database and associate it with the server group.	• Database tables

Data source configuration roadmap

You configure data sources in Opportunity Detect as described in this roadmap. When you have finished this process, you can build, deploy, and run trigger systems, as described in the *IBM Opportunity Detect User's Guide*.

You can access all of the configuration tools mentioned here by navigating to the **Settings > Detect Settings** page.

You must have appropriate permissions to access the configuration tools. The following built in roles allow the user to access these tools.

- **OpDetectAdmin**
- **OpDetectTestDesigner**
- **OpDetectProductionDesigner**

If you use a custom role, it must have either of the following permissions.

- **Run for testing**
- **Run for production**

Table 4. Data source configuration roadmap

Step	Where to find details
Set up your State History and Outcome database tables. Required.	"Setting up State History and Outcome tables" on page 13
On the Audience Levels page, configure audience codes. Required.	"Configuring audience level codes" on page 14
On the Named Value Lists page, create display names for pre-defined values in your transaction and profile data sources. Optional.	"Configuring named value lists" on page 16
On the Data Sources page, identify the fields in your transaction and profile data sources and specify display names. Required.	"Defining data sources" on page 17
On the Server Groups page, define database connections, data source connectors, and server groups, and map connectors to data sources. Required.	"Configuring server groups and data source connections" on page 19

Table 4. Data source configuration roadmap (continued)

Step	Where to find details
Set up your transaction and profile data feeds. Required.	• If you use the flat file format, ensure that the files meet the requirements for data used in Opportunity Detect. – "Naming requirements for profile and batch transaction files" on page 31 – "Data requirements for profile and batch transaction files" on page 32 – "Profile data" on page 33 • If you use the web service connector for transactions, see the *IBM Opportunity Detect Administrator's Guide* for details about the Java classes that must be developed to use the Web Service data source connector.

Setting up State History and Outcome tables

Use this procedure to edit the SQL scripts provided for creating your State History and Outcome tables and to run the scripts against your database.

Before you begin

You must set up a data source connection on the Server Groups page for each database where you create these tables.

About this task

You do not need to use this procedure to set up Outcome tables if you use the Opportunity Detect Web Service data source connector for your Outcome data, because the data is sent to the Java class you develop.

Procedure

1. Open the SQL scripts in a text editor and edit them as described in the comments in the files.

   The scripts are located in the `database/`*`database type`*`/RunTime` directory under your Opportunity Detect run time installation. Edit the following scripts.

   - `OutcomeTable.sql`-Use this script to create the Outcome table when you use the Outcome type of data source connector.
   - `ExpandedOutcome.sql`-Use this script to create the two Outcome tables when you use the Expanded Outcome type of data source connector.
   - `StateTable.sql`-Use this script to create the State History table or tables.

   You replace variables with the names you want to give to the tables in the database. Comments in the scripts provide instructions.
2. Run the scripts against the database you have configured to hold your State History and Outcome data.

Audience level codes

Audience codes represent the various audience levels defined in your data. Opportunity Detect groups data sources based on the audience code associated with them.

About audience levels

An audience level is composed of a key or database table field that uniquely identifies a member of that audience level. For example, your organization could use the audience levels Household, Customer, and Account. Each of these levels represents a different view of your marketing data.

Audience levels are typically organized hierarchically. Using the examples above, you might have the following hierarchy.

- Household is at the top of the hierarchy, and each household can contain multiple customers.
- Customer is next in the hierarchy, and each customer can have multiple accounts.
- Account is at the bottom of the hierarchy.

Other, more complex examples of audience hierarchies exist in business-to-business environments, where audience levels may need to exist for businesses, companies, divisions, groups, individuals, accounts, and so on.

These audience levels may have different relationships with each other, for example one-to-one, many-to-one, or many-to-many. By defining audience levels, you allow these concepts to be represented within Opportunity Detect, so users can manage the relationships among these different audiences for targeting purposes. For example, although there might be multiple customers per household, you might want to limit contact to one customer per household.

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

How audiences are associated with data sources

When you define your data sources on the Data Sources page, you specify the fields you want to use in Opportunity Detect. When you specify an audience ID field, you associate that field with an audience level defined in Opportunity Detect.

Audience codes for internal use

Opportunity Detect includes a pre-defined audience code: `n`. The `n` code is used internally by the system and is read-only. You must create additional audience codes to represent the audience levels in your data.

Also, the letter i is reserved for real time mode, and signifies the audience used for the audience level used by Interact when integration is enabled. You should not use i when configuring your audience levels for batch mode.

Configuring audience level codes

Use this procedure to specify audience level codes in Opportunity Detect.

Procedure

1. Log in to Opportunity Detect and go to the **Settings > Detect Settings > Audience Levels** page.
2. Click **Add Audience Level**.
3. Give the audience level a name and a code.

The audience code must be a single alphanumeric ASCII character, lower case (0-9, a-z except i).

4. Click **Save and Close**.

Named value lists enable display names for predefined field values

When users of Opportunity Detect work with components to build trigger systems, they can select data fields from drop-down lists in the component editors. If a data source includes a field with predefined values, you can create a named value list to assign display names to these values.

The display names are what the user sees when working with component editors. Display names make the component-building process more intuitive.

Named value lists are reusable. For example, you might have several transaction data sources that have a Boolean field where the predefined values are `0` and `1`, where `1=true` and `0=false`. In this case, you can create a named value list that you associate with these fields in several transaction data sources.

Named value list example

For example, you might have a field in a data source that is named `FREQUENCY`. This field contains three predefined values, which have the following significance.

- `F` for customers who interact with your company frequently
- `A` for customers whose interaction frequency is average
- `I` for customers who interact with your company rarely

To help users understand what these values mean, you can map each value to a display name. You might create a named value list called `Frequency` and map the values as follows.

Table 5. Mapping data to display names

Value	Display name
F	Frequent customer
A	Average customer
I	Infrequent customer

After you map the values, you can associate the list with the field when you define the data source. Then, when a user selects this field in a component editor, the drop-down list that contains the predefined values shows the display names, which are more meaningful to the user than the actual values in the data source.

Predefined lists

Three lists are present when Opportunity Detect is installed, for use in the component editors. The values are mapped to display names as follows, but you can modify the values as needed to reflect your data.

Table 6. Display name mapping in predefined lists

List name	Display name mapping
NUMERIC_YES_NO	• Yes = 1 • No = 0

Table 6. Display name mapping in predefined lists (continued)

List name	Display name mapping
DAY_WEEK_MONTH	• Day = day • Week = week • Month - month
HIGH_MED_LOW	• High = H • Medium = M • Low = L

Configuring named value lists

Use this procedure to create named value lists in Opportunity Detect.

Before you begin

Obtain the following information about your data.

- Pre-defined values - You need these when you create the named value list items in this procedure.
- Name of the field in your data source that contains predefined values - You need this when you associate the named value list to the appropriate field in your data source.

Procedure

1. Log in to Opportunity Detect and go to the **Settings > Detect Settings > Named Value Lists** page.
2. Click **Add List**.
3. In the Properties panel, give the list a name and description, and specify the data type.

   The data type must match the data type of the field in your data source.
4. In the List Items panel, enter a display name and the predefined value that exists in your data source, and click **Add List Item**.

   Repeat this step for each predefined value for which you want to create a display name.
5. Click **Save and Close**.

What to do next

When you specify transaction and profile fields on the Data Sources page, associate the named value list with the appropriate field.

Deleting and modifying named value lists

Use this procedure to delete or modify named value lists in Opportunity Detect.

Before you begin

Before you can delete a list or a list item, you must either delete all of the components that use any of the list items, or edit those components to remove the reference to the list item.

The components that use items in a list are shown in the List Item and Component Association panel.

About this task

The predefined lists can be modified, but they not be deleted.

Procedure

1. Log in to Opportunity Detect and go to the **Settings > Detect Settings > Named Value Lists** page.
2. To delete a list, select the list you want to delete, click **Delete List**, and click **OK**.
3. To delete a list item, select the list to which the item belongs, select the item you want to delete, click **Delete List Item**, and click **OK**.

Defining data sources

Use the **Data Sources** page to a define each transaction and profile data source that you want to use in Opportunity Detect.

About this task

When you define the transaction and profile data sources that you want to make available in workspaces, you do the following.

- Specify the type of data source and a display name for the data source.
- Specify every field that you want to make available for use in building trigger systems.
- Optionally, associate named value lists with fields that have predefined values.

Transaction and profile data can not include any non-printing or meta characters.

Procedure

1. Log in to Opportunity Detect and go to the **Settings > Detect Settings > Data Sources** page.
2. Click **Add Source**.
3. In the Properties panel, give the data source a name and description and specify the content type.

   The content type describes how the data is used in Opportunity Detect.
4. In the Fields panel, click **Add Fields**, complete the fields in the Add a Data Source Field window and click **OK**.

   Repeat this step for each field that you want to use as a data source.

   For every transaction data source, you must define an audience field and a date field.

   Profile data sources require only an audience field. Only one profile data source per audience level is supported.

   For profile data sources in database tables, the value you enter in the **Name** field must be the same as the corresponding column name in the database.

   Optionally, select a named value list to associate with the field, if the field contains predefined values.
5. Click **Save and Close**.

Fields on the Data Sources page

You define data sources on the Data Sources page. If you are unsure of how to complete the fields, use the information provided here.

Table 7. Fields on the Data Sources page

Field	Description
Properties	
Content Type	Select one of the following options. • **Transaction** - Customer transactions. At least one Transaction data source is required to run a workspace. • **Profile** - Data such as address, age, and gender in an external database table.
Name	Enter a descriptive name for the data source. • For profile data sources in database tables, the name must be the same as the corresponding column name in the database. • For transaction data sources, the name does not have to match the field name in the actual flat file or web service message.
Description	Provide a description of the data source.
Add Field	
Name	Provide the physical name of the field in the source database table or flat file. Data source field names should not be assigned so that one name can be found within another. For example, the name `date` can be found within the name `firstdate`. Field names cannot use spaces, hyphens, or periods. Restrict the field names to letters, numbers, and underscores.
Display Name	Provide a descriptive name. This appears in component editors.
Description	Provide a description of the data source field.
Field Type	Select the attribute type from the following options. The options available depend on the Content Type selected for the data source. • Audience Id-Select this for audience level fields. • Audience Level-If you select Audience Id as the Field Type, select the audience level you want to associate with this field. • Profile Attribute-Choose this type for profile fields that are not audience or date fields. • Transaction Attribute-Choose this type for transaction fields that are not audience or date fields. • Transaction Date-Choose this type for date fields. One date field is required for each transaction data source.
Audience Level	If you selected Audience Id as the Field Type, enter the audience level code that applies for the field.

Table 7. Fields on the Data Sources page (continued)

Field	Description
Data Type	Select the type of the data in the field. The options available depend on the Field Type selected for the field, and also on the custom data types defined on the Custom Data Types page, if any. The following are the default options. • Boolean • Currency • Date • Double • Integer • String The application supports profile data sources in database tables in a variety of database types. The data types shown in this field are generic types that may be different in your database. The system converts the type you select into the correct data type for your database.
Name Value List	Named value lists with a data type that matches the field you are working with are displayed here. If the field contains predefined values, you can associate it with a named value list.

Deleting and modifying data sources

Use this procedure to delete or modify data sources in Opportunity Detect.

Before you begin

If you want to edit a data source that is being used by a component, you must either delete the components that use the data source, or edit those components to remove the reference to the data source.

The components that use the data source are shown in the Source Field and Component Association panel.

Procedure

1. Log in to Opportunity Detect and go to the **Settings > Detect Settings > Data Sources** page and select the data source you want to work with.
2. To delete the whole data source, click **Delete Source** and click **OK**.
3. To add a field, click Add Field and click **OK**.
4. To delete a field, select the field, click **Delete Field** and click **OK**.

Configuring server groups and data source connections

Use this procedure to define server groups and connect them to data sources you have configured.

About this task

Server groups are comprised of the following.

- Streams servers that are set up during installation
- Database connections that you create

- Data source connectors that you create and map to data sources

Typically, organizations use different server groups for their development, test, and production environments. Development and test server groups often share Outcome and State History tables, while the production server group normally has its own, exclusive Outcome and State History tables.

Procedure

1. Log in to Opportunity Detect and go to the **Settings > Detect Settings > Server Groups** page.
2. The first time you use the Server Groups page, and whenever you add a new server, click **Synchronize with changed streams configuration** on the Servers tab to refresh the list of servers.
3. On the Database Connections tab, configure a connection for every database containing the following tables that you plan to use with Opportunity Detect.
   - Run time tables
   - Outcome tables
   - State History
   - Profile

   You need the database name.
4. On the Data Source Connectors tab, configure connectors for your transaction, profile, Outcome, and State History data.

   You do not have to set up a connector for your run time tables.

   For transaction and profile data in file format, you specify the encoding, locale, and currency format used in the file. Opportunity Detect constructs the required file name, and you must ensure that your data files use this name.
5. On the Server Groups tab, select a server group to open a panel with four tabs. Configure server groups as follows.
   a. On the Properties tab, select an available Streams server and instance, and specify whether the server group is used for production.

      You can also view a list of workspaces that have deployment configurations associated with the server group.
   b. On the Servers tab, double-click in the **# of Engines** column to set the number of Opportunity Detect engines to run on this machine.
   c. On the Database tab, select a run time database and select the database connections you want to make available for table connector mapping.
   d. On the Data Source Mapping tab, map transaction and profile data sources and your Outcome and State History tables to data source connectors.

      Table connectors and flat file connectors enable the system to connect to your Outcome and State History database tables and your transaction and profile files.

Related concepts:

"About data source connectors" on page 28

Related reference:

"Fields on the Server Groups page" on page 21

Fields on the Server Groups page

If you are unsure of how to complete the fields on the Server Groups page, use the information provided here.

Table 8. Server Groups page: Servers tab

Field	Description
Synchronize with changed streams configuration	Click to refresh the list of servers and stream instances.

Table 9. Server Groups page: Database Connections tab.

Tip: If a database connection error occurs, verify that the information entered in these fields is correct.

Field	Description
Add	Click to open a panel where you can add a database connection.
Name	Enter a descriptive name for this database.
Database Type	Select a database type from the drop-down list of supported databases.
Database Name	Enter the name of the database as shown in your database management client. This must exactly match the name as shown in the client. Case-sensitive.
Server Name	Enter the fully qualified name or IP address of the machine that hosts the database server. For example, machine.mycompany.com.
Port	Enter the port on which the database listens. The default port for DB2 is 50000.
User Id	Enter the user name of the database account you want Opportunity Detect to use to access to this database.
Password, Confirm Password	Enter the password for the account you entered in the **User ID** field.

Table 10. Server Groups page: Data Source Connectors tab

Field	Description
Add	Click to select the type of connector to add. The options are: • `File` • `Queue` • `Table`
Batch file connector	
Name	Enter a descriptive name for the file connector.
File Name	Click the link in this field to open a pop-up window, and complete the following fields. • Name - Enter the base file name for this flat file. Permitted characters are ASCII letters, numbers, and underscore. Do not use spaces or any other special characters. • Audience Level - Select from a drop-down list of available audience levels. Opportunity Detect constructs a complete file name from the base name you provide. The name of your file must match the name that Opportunity Detect constructs.

Table 10. Server Groups page: Data Source Connectors tab (continued)

Field	Description
Description	Enter a description of this file.
Delimiter	Select a delimiter for data within records. Options are: • **Pipe** • **Comma** • **Semicoloin** • **Tab**
File Encoding	Select the encoding used in the file from the drop-down list. Options are: • Chinese Traditional (Big 5) • Unicode (Little endian) • Unicode (Big endian) • Western European (ISO) • Central European (ISO) • Latin 3 (ISO) • Latin 9 (ISO) • Korean (EUC) • Chinese (GB 18030) • Unicode (UTF-7) • Unicode (UTF-8)
Date Locale	Select from a drop-down list of supported locales. Options are: • English (United States) • Chinese (China) • English (United Kingdom) • French • German • Italian • Japanese (Japan) • Korean • Portuguese (Brazil) • Russian • Spanish • Thai
Currency	Select from a drop-down list of supported locales. Options are the same as for **Date Locale**.
Queue connector **Note:** For additional information on setting up queue data source connectors, see Chapter 3, "Real time processing in Opportunity Detect," on page 9.	
Name	A descriptive name for the queue connector.
Queue type	Select the type of queue server you are using: **Active MQ**, **WebSphere MQ**, or **Rabbit MQ**.
Queue Name	Name of the queue as configured on the queue server. Active MQ and WebSphere MQ only.
Connection URL	An example value appropriate for the queue type you selected is provided. Active MQ and WebSphere MQ only.
Connection Factory	An example value appropriate for the queue type you selected is provided. Active MQ and WebSphere MQ only.

Table 10. Server Groups page: Data Source Connectors tab (continued)

Field	Description
Connection Factory Identifier	An example value appropriate for the queue type you selected is provided. Active MQ and WebSphere MQ only.
Exchange Name	The name of the direct exchange configured on the Rabbit MQ server. Rabbit MQ only.
Host Name	The host name of the Rabbit MQ server. Rabbit MQ only.
Port	The port of the Rabbit MQ server. Rabbit MQ only.
User Id	The user name for the account that the system uses to access the queue server. If this value changes after deployment, you must redeploy any workspaces that use this connector.
Password	The password for the account that the system uses to access the queue server. If this value changes after deployment, you must redeploy any workspaces that use this connector.
Confirm Password	The password for the account that the system uses to access the queue server.
Description	A description for this queue connector.
Sharable	Select this check box if you want to be able to use this queue connector on more than one server group. Active MQ and WebSphere MQ only. Rabbit MQ connectors are not sharable because with Rabbit MQ the input is consumed in a round robin pattern.
Real time file connector Fields are the same as for Batch file connector, with the following additional fields.	
File Name	Use a POSIX regular expression to specify the name pattern for the files in the input directory. The input directory is specified on the Properties tab of a deployment.
Data source	Select a data source defined on the **Settings > Detect Settings > Data Sources** page. The **Field definition details** grid is populated with the fields defined for the data source.
Enable Bloom filter	The Bloom filter eliminates duplicate data based upon user-defined filter fields, as follows. • **Time interval** - Controls the time span over which the filter looks for the duplicate data, based on the timestamps of the records.. • **Maximum unique** - Sets the maximum number of unique records to store in memory. Valid choices are between 1-99999999999999. • **Probability** - Sets the probability of finding duplicate records, to minimize the number of false positive matches from the Bloom filter. Valid values are any decimal value greater than 0 and less than 1. When a Bloom filter is applied, the ID and timestamp fields are automatically selected as filter fields and cannot be unselected. You can choose to add additional fields as part of the filter criteria.
Field definition details	Define the start position and length of each field, and, if the Bloom filter is enabled, whether to apply the Bloom filter to the field.

Table 10. Server Groups page: Data Source Connectors tab (continued)

Field	Description
Table Connector	
Name	Enter a descriptive name for this table connector. It is a good practice to align this name with the name of the database you plan to associate with the connector.
Type	Select a type from the drop-down list. The options are: • `State` • `Profile` • `Outcome` • `Expanded Outcome`
Table Name	Enter the name of the table as shown in your database management client. This must exactly match the name as shown in the client. Case-sensitive.
Description	Enter a description of the table.
Sharable	Select this check box if you want to be able to use this table connector on different server groups.

Table 11. Server Groups page: Server Groups tab

Field	Description
Add	Click to open a panel where you can define a server group.
Properties tab	
Name	Enter a descriptive name for the server group.
Stream Instance	Select a stream instance ID. See your administrator if you are not sure which one to select.
Usage	Enter a brief description of how this server group is used. For example, Production or Development.
For Production	Select this check box if you want to restrict user's ability to run workspaces on this server, based on their permissions.
Servers tab	
Fully Qualified Name	Double-click in the **# of Engines** column to set the number of Opportunity Detect engines to run on this machine.
Database tab	
Select Runtime database connection	The run time database holds the tables where your run, run details, and run status data is stored. Select the run time database that you want this server group to be able to access.
Select database connections for table connector mapping	The database connections you check here are the ones that are available when you map a table data source to a connector in the Table Data Source Connector Mapping window.
Data Source Mapping tab	
List of datasources	Click the name of a data source to open a pop-up window where you can map the data source to a connector. Only sharable table connectors are available for mapping in the server group page. A connector can be mapped to only one data source.
Data Source Connector Mapping tab	

Table 11. Server Groups page: Server Groups tab (continued)

Field	Description
Data Source Name	This is a read-only field that contains the name of the data source you clicked to open this window.
Connector Type	Select from a list of data source connector types.
Connector	Select from a list of previously configured connectors of the type selected.
Database connection	For table connectors only, select from a list of previously configured database connections.

Related concepts:

"About data source connectors" on page 28

"About deployment configurations" on page 35

Chapter 7, "Queue connectors for input and output in Opportunity Detect," on page 57

Chapter 8, "Web Service connectors for input and output in Opportunity Detect," on page 69

"Audience level codes" on page 13

Related tasks:

"Configuring server groups and data source connections" on page 19

"Configuring audience level codes" on page 14

Regular expressions in Opportunity Detect

In Opportunity Detect, you use regular expressions in two situations.

- When you create a Real time file connector in the Server Groups page, you use a regular expression to match the pattern used in file names.
- When you create a Boolean expression in a component and you select the **Like** operator, you can use a regular expression to set the criteria for comparison.

Opportunity Detect uses the Streams standard toolkit for matching regular expressions. Opportunity Detect supports the POSIX extended regular expressions standard.

The regular expression must conform to the Streams Processing Language requirements, described here: https://www-01.ibm.com/support/knowledgecenter/SSCRJU_3.2.0/com.ibm.swg.im.infosphere.streams.spl-language-specification.doc/doc/primitivetypes.html

Take care that the pattern you specify exactly matches your intent. Some level of testing is always advisable to verify that your patterns are actually matching the required expressions. You can use a trial and error process to design patterns, starting with low complexity and changing them bit by bit to achieve the required result. Pay particular attention to escaping backslashes.

Special characters

Here is a summary of special character usage in POSIX regular expressions.

- Period (.) : Matches any character.

- Anchors (^, $) : The (^) anchor defines the start of the expression, and the ($) anchor defines the end of the expression.
- Asterisk (*) : A quantifier that matches a single character or group of characters any number of times.
- Plus (+) : A quantifier that matches a single character or a group, one or more times.
- Question mark (?) : A quantifier that represents optional items.

Bracket expressions

A bracket expression represents a class of characters, any one of which could be a match a single character. For example `[a-c]` is a bracket expression that will match any of the characters a, b, or c. For example: the regex `[a-c]+` will match aaa, abc, ca, etc; or any string containing a sequence of at least one character from the set a, b, or c followed by any number of characters also from that set.

There are other forms of bracket expressions. For example, `[a-c]` could be also specified as `[abc]`. Within a bracket expression, there are collating elements. It has the form `[.col.]`. (There might be other forms.) A collating element is a character or group of characters that act as a single character in a bracket expression. For example, if `[.ae.]` is a collating element, then it can be used within a bracket expression `[[.ae.]bc]`, which states: match any of the characters "ae", b, or c. In other words, it forces ae to be treated as a single character.

Table 12. Character classes

POSIX	Description	ASCII
`[[:alnum:]]`	Alphanumeric characters	`[a-zA-Z0-9]`
`[[:alpha:]]`	Alphabetic characters	`[a-zA-Z]`
`[[:blank:]]`	Space and tab	`[ \t]`
`[[:cntrl:]]`	Control characters	`[\x00-\x1F\x7F]`
`[[:digit:]]`	Digits	`[0-9]`
`[[:graph:]]`	Visible characters (that is, anything except spaces, control characters, etc.)	`[\x21-\x7E]`
`[[:lower:]]`	Lowercase characters	`[a-z]`
`[[:print:]]`	Visible characters and spaces (that is, anything except control characters, etc.)	`[\x20-\x7E]`
`[[:punct:]]`	Punctuation and symbols	`[!"#$%∓'()*+,-./:;<=>?@[\]^_`{}~]`
`[[:space:]]`	All whitespace characters, including line breaks	`[ \t\r\n\v\f]`
`[[:upper:]]`	Uppercase letters	`[A-Z]`
`[[:xdigit:]]`	Hexadecimal digits	`[A-Fa-f0-9]`

Quantification

The question mark makes the preceding token in the regular expression optional. For example, colou?r matches both colour and color.

The star (*) tells the engine to attempt to match the preceding token zero or more times. The plus sign (+) tells the engine to attempt to match the preceding token one or more times.

An additional quantifier allows you to specify how many times a token can be repeated. The syntax is {min,max}, where min is zero or a positive integer indicating the minimum number of matches, and max is an integer equal to or greater than min indicating the maximum number of matches. If the comma is present but max is omitted, the maximum number of matches is infinite.

For example:

- `{0,1}` is the same as `?`
- `{0,}` is the same as `*`
- `{1,}` is the same as `+`

Omitting both the comma and max tells the engine to repeat the token exactly min times.

You could use `\b[1-9][0-9]{3}\b` to match a number between 1000 and 9999. `\b[1-9][0-9]{2,4}\b` matches a number between 100 and 99999. Notice the use of the word boundaries.

Grouping

Single characters, or expressions matching single characters, enclosed in parentheses (round brackets), are treated as a regular expression matching a single character. That is, quantification and other rules apply to the group in the parentheses as a whole.

Alternation

Two regular expressions separated by the special character vertical-line (`'|'`) match a string that is matched by either.

For example, the regular expression "a((bc)|d)" matches the string "abc" and the string "ad".

Single characters, or expressions matching single characters, separated by the vertical bar and enclosed in parentheses, are treated as a regular expression matching a single character.

Example for file name matching

You might create the following regular expression to match timestamp suffixed file names used with the Real time file connector.

```
Detect\.a\.trans\.[0-9]{8,14}
```

This expression matches file names with the common prefix `Detect.a.trans` and ending with timestamp digits of length greater than 8 and less than 14. This is done because file names can have 8 digits for the basic date (4 for year, 2 for month, 2 for date) and 6 extra digits for more granular timestamps (hh:mm:ss).

```
Detect.a.trans.20100901
Detect.a.trans.20100908
Detect.a.trans.20100922
Detect.a.trans.20101001
Detect.a.trans.20101008
Detect.a.trans.20101022
Detect.a.trans.20101201
Detect.a.trans.20101208
```

```
Detect.a.trans.20101222
Detect.a.trans.20101222
Detect.a.trans.20101223040506
Detect.a.trans.20101223033240
```

Useful links for POSIX regular expressions

- OpenGroup POSIX regular expression specification:
  http://pubs.opengroup.org/onlinepubs/009696899/basedefs/xbd_chap09.html
- http://www.regular-expressions.info/posix.html
- Wikipedia regular expressions:
  https://en.wikipedia.org/wiki/Regular_expression#POSIX_basic_and_extended
- Wikibooks POSIX regular expressions:
  https://en.wikibooks.org/wiki/Regular_Expressions/POSIX_Basic_Regular_Expressions

About data source connectors

Data source connectors enable the system to connect to the transaction, profile, lookup, State History, and Outcome data sources that you have defined. The types of data source connectors are described in this section.

If you add or change fields in a data source or data source connector, you must redeploy any workspaces with data source connectors that use the changed data source.

Note: For Outcomes, the data source connector you use determines the format of the Outcome data.

Batch file connectors

Use Batch file connectors to connect to transaction and profile data that is in file format and that is processed in batches.

Real time file connectors

Use Real time file connectors to connect to transaction data that is in fixed width file format and that is updated frequently. This connector is particularly useful for Call Data Records (CDRs) used by the telco industry. The CDR must first be transformed from binary to ASCII fixed width format.

Opportunity Detect reads these files in real time. You configure your automation system to place the properly formatted file in an input directory that you specify when you create the deployment configuration. After Opportunity Detect consumes the data, the files are automatically moved to another directory that you specify for precessed files.

The input and output directories must be on the runtime server, and must have Streamsadmin permission.

When you create the Real time connector, you enter the start and length of each field. The values must be a positive integer greater than 0.

You can choose to have the connector use a Bloom filter on the data. The Bloom filter eliminates duplicate data based upon user-defined filter fields. Data rejected by Bloom filter is placed under the deployment ID folder: `/home/streamsadmin/`

`OpDetection/deploy/`*`Deployment ID`*`/current/data` You should move this rejected duplicate data on a regular basis, to conserve disk space on the runtime server.

If you have more than one deployment using the Real time file connector, ensure that a different folder is designated for the transaction files for each deployment, to prevent an condition where the file is moved before one of the deployments is finished with it.

Opportunity Detect does not support live updates to files in the input directory used with the real time file connector. Also, you should not use the same input directory for multiple workspaces, as this could lead to undesired behavior.

Table connectors

Use table connectors to connect to the following data sources.

- Profile data that is in database table format
- All State History tables
- Outcome data that you want to be written to database tables

There are four types of table data source connectors.

State

Used to connect to your State History tables.

Profile

Used to connect to your profile data that is in database table format.

Outcome

Used when you want Outcome data to be stored in database tables in the XML format used by previous versions of Opportunity Detect.

Expanded Outcome

Used when you want Outcome data to be stored in database tables in a form that external systems or IBM Campaign can use more easily than the format provided by the Outcome connector.

A table connector can be mapped to only one data source.

Web Service

You can use the Web Service connector to connect to transaction and Outcome data coming from or being sent to the Opportunity Detect web service.

If you use the web service for transaction data and you also use profile data, the profile data must be in database table format.

You do not have to create this type of connector. It exists in the system by default.

For details about the Java classes that must be developed to use the Web Service data source connector, see "Web Service data source connector for input and output in Opportunity Detect" in the *IBM Opportunity Detect Administrator's Guide*.

Queue

You can use this connector for real time operation in conjunction with a supported queue server.

For additional information on setting up Queue data source connectors, see "Real time processing in Opportunity Detect" in the *IBM Opportunity Detect Administrator's Guide*.

Default TCP Connector

Used by the system with IBM Interact Advanced Patterns when integration with IBM Interact is implemented.

You do not have to create this type of connector. It exists in the system by default.

The sharable option

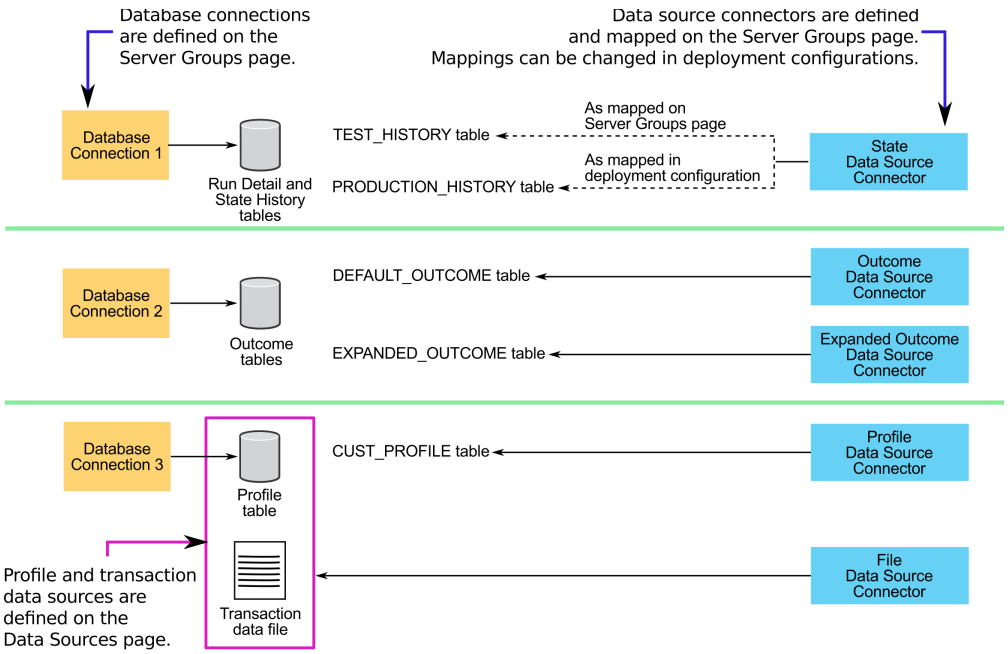
You have the option to make table and queue connectors sharable. The sharable feature works as follows.

- A sharable connector can be used in more than one server group.
- Sharable connectors can be mapped on the Data Source Mapping tab of the Server Groups page and on the Deployment tab of a workspace. Table and queue connectors that are not sharable can be mapped only on the Deployment tab of a workspace.

For example, you might want to have connectors for Outcome and State History tables that are not shared and can be used only by the deployment configuration used for production. You might also want sharable table connectors so you can map them to the same Outcome and State History tables for all test workspaces.

Example of data source connector configuration

The following diagram illustrates a possible data source configuration, showing the relationship between database connections, data source connectors, and the data sources used in Opportunity Detect.



Related concepts:

Chapter 3, "Real time processing in Opportunity Detect," on page 9

Chapter 8, “Web Service connectors for input and output in Opportunity Detect,” on page 69
“About deployment configurations” on page 35
Chapter 7, “Queue connectors for input and output in Opportunity Detect,” on page 57
Chapter 6, “Outcome data in Opportunity Detect,” on page 47

Related tasks:

“Configuring server groups and data source connections” on page 19

Related reference:

“Fields on the Server Groups page” on page 21
“Fields and buttons on the Deployment tab” on page 41

Naming requirements for profile and batch transaction files

As part of the process for setting up server groups, you can configure a Batch file connector on the Data Source Connectors tab on the Server Groups page. The system uses the name you specify to generate a file name.

Ensure that the names of your batch transaction and profile feed files exactly match the name shown on the Data Source Connectors tab on the Server Groups page.

When you define Batch file data source connectors for server groups, you supply a name. The system expects the names of batch transaction and profile feed files to contain this name plus additional information, as follows. In this example, *data_source_name* is the name you enter when you define the data source connector.

`Detect.data_source_name.date`

You must use this name format for the transaction and profile feed files that correspond to the data sources that you configure in Opportunity Detect.

Permitted characters for the name you enter are ASCII letters, numbers, and underscore. Do not use spaces or any other special characters

Date portion of the name

The *date* portion of the batch file name must have this format: `YYYYMMDDhhmmss`

Names of batch files must contain a date.

The date determines the order in which the files are processed. Files with the same timestamp are processed simultaneously. If you use a profile, you must create one for each set of files with a different timestamp.

During ramp-up, you might process multiple transaction files with different dates in a single run. If you are using profile files, you must include different profile files named with dates that match the date in the names of each of the transaction files.

How batch transaction files are processed

Opportunity Detect uses the *data_source_name* portion of the file name to identify multiple instances of a transaction data source.

You might have a series of batch transaction files, all with exactly the same fields, using different timestamps in their names. Every time you run a trigger system that uses this data, a file with a different date stamp and/or timestamp would be used.

Data requirements for profile and batch transaction files

Profile and batch transaction files have a specified format.

Format requirements

Batch files must have the following format.

- One header that contains the names of the fields in the file. The names must exactly match the names defined in the data source to which the Batch file connector is mapped.
- The first field must be the audience ID.
- The file must have UNIX style newline characters.
- All record lines must use a delimiter between fields, but not before the first field in the line or after the last field in the line.

  The following delimiters are allowed.
  - pipe (vertical line: |)
  - comma
  - semi-colon
  - tab
- 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.
- The delimiter must not be used within data fields.

Date field requirement

For batch transaction files only, one field must be a timestamp field, expressed in milliseconds.

Audience ID field requirement

One field in all data sources must be an ID field used for an audience level, and this must be the first field in each record in the file.

Sort order requirement

The records in a batch file must be sorted by audience ID first, then by transaction timestamp.

Additional considerations

Note the following considerations.

- Opportunity Detect cannot 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 use NULL as a condition in comparisons.
- Date fields must be in system-readable date format even if you want those dates to be empty (NULL). You can represent NULL dates as either very old dates or very future dates .
- The date format in date fields must match the date format that is associated with the date locale used in the file connector.
- In distributed environment, all runtime servers must be able to access the profile and transaction files.

Batch transaction and profile file examples

Note: The following examples are formatted with constant widths for readability. Actual batch transaction and profile files must not use constant widths.

Here is an example of a simple transaction file.

```
ID    |NAME    |CALLED_NUMBER|CALL_LENGTH|TRAN_DATE_TIME
001234|David   |732-123-4567 |15         |2012-02-10 09:12:33
001234|David   |732-111-5555 |48         |2012-02-10 10:11:50
002941|Jeremiah|732-777-8888 |40         |2012-02-10 11:22:44
005555|Anthony |732-333-4444 |27         |2012-02-10 03:01:02
005555|Anthony |732-32-8945  |121        |2012-02-10 10:12:30
005555|Anthony |973-597-0022 |2          |2012-02-10 19:00:21
006789|Tom     |732-111-2222 |4          |2012-02-10 06:54:01
```

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
```

Profile data

The following requirements apply to all profile data sources used in trigger systems.

- Profile data is not required. However, if profile data is not defined, all the data used in a trigger system (including any required profile data) must be included in the transaction data.
- When profile data is defined, for every customer represented in your transaction file, there must be a single record for that customer in your associated profile.

  When a transaction file is associated with a profile flat file, the system ignores transaction records without an associated profile record, and ignores profile records without an associated transaction record. In both cases, these orphan records affect processing efficiency.
- When profile files are used, a profile file must exist for each set of transactions with the same date suffix. The date suffix of the profile file must match that of the transaction files. A profile file with a different date suffix cannot be used.

Data requirements for real time transaction files

Files used with the Real time data source connector must be in fixed width format. The following requirements apply.

- One field must be an ID field used for an audience level, and this must be the first field in each record in the file.
- One field must be a timestamp field.
- Spaces must be used for padding where necessary to make a field of the length specified in the Real time file connector. Padding may be either before or after the data.

Date and currency formats for supported locales

Opportunity Detect supports a variety of locales at the data level.

The following table shows the expected format for dates and currency in the supported locales.

Table 13. Date and currency formats by locale

Locale	Currency format	Date format
de_DE (German)	123.456,78	dd.MM.yyyy HH:mm:ss
en_US (English - United States)	123,456.78	MMM d, yyyy h:mm:ss a
en_GB (English - United Kingdom)	123,456.78	dd-MMM-yyyy HH:mm:ss
es_ES (Spanish)	123.456,78	dd-MMM-yyyy H:mm:ss
fr_FR (French)	123 456,78	d MMM yyyy HH:mm:ss
ko_KR (Korean)	123,456.78	yyyy. M. d a h:mm:ss
ja_JP (Japanese - Japan)	123,456.78	yyyy/MM/dd H:mm:ss
pt_BR (Portuguese - Brazil)	123.456,78	dd/MM/yyyy HH:mm:ss
zh_CN (Chinese - China)	123,456.78	yyyy-M-d H:mm:ss
it_IT (Italian)	123.456,78	d-MMM-yyyy H.mm.ss
ru_RU (Russian)	123 456,78	dd.MM.yyyy H:mm:ss

Chapter 5. Deploying and running workspaces

You can create and run deployment configurations and view deployment and batch run history on the Workspaces page.

- On the Deployment tab, you can create deployment configurations for workspaces, deploy the configurations, stop and restart deployments, and view deployment history.
- On the Batch Run tab, you can manage run parameters, run deployed workspaces, see the status of the most recent batch runs, and stop and start ongoing runs.
- On the Batch History tab, you can see details about past runs, filtered by deployment and a date range.

About deployment configurations

Before you can run a workspace to process data, you create a deployment configuration for it and then deploy it.

When you deploy a configuration, the workspace is compiled into a Streams application and sent with all of its component logic to the Streams server.

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

Each workspace has its own configuration deployment, and multiple configuration deployments can be running simultaneously.

Server group availability in deployment configurations

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

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

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

Option to use different configurations for test and production

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

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

the test data from the production data in your State History and Outcome tables.

Required permissions

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

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

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

Redeployment requirement when data sources change

If you add or change fields in a data source or data source connector, you must redeploy any workspaces with data source connectors that use the changed data source.

Related concepts:

Chapter 8, "Web Service connectors for input and output in Opportunity Detect," on page 69

"About data source connectors" on page 28

Related reference:

"Fields on the Server Groups page" on page 21

Data source connector mapping

Data source connectors can be mapped to different data sources in each deployment configuration, which provides flexibility for testing and ramp-up operations.

Default mappings between data source connections and database tables are set when server groups are configured. However, in deployment configurations, you can change the default mappings for a selected server group. Your changed mappings apply only within the deployment configuration and do not affect the global, default mappings for the server group.

Mapping for testing

Suppose you have a server group named Test. You might want to use one set of State History and Outcome tables with the Test server group in one workspace, and another set of State History and Outcome tables with the Test server group in another workspace.

You can do this by changing the mapping of the data source connector on the Data Source Mapping tab for the deployment configuration.

On the other hand, you might want to use the same State History table for several different test workspaces, to reduce the number of State History tables and data source connectors that you need to create.

Mapping for ramp-up

In Opportunity Detect, ramp-up is the process of accumulating State History data for Container and Pattern components. This enables the system to produce meaningful outcomes quickly, rather than waiting for the history to accumulate over time.

For example, trigger system designers often need to introduce a new trigger system into a production workspace that already includes multiple trigger systems. Suppose that the production workspace consists of 10 trigger systems that have been in production for several months. When a new trigger system is introduced, its history can be ramped up by running it in its own workspace against historical transactions, using the same State History table used for the production workspace.

Because the new trigger system is run in isolation, running it against historical transactions does not affect any of the history of the 10 triggers already in production. After ramp-up, the new trigger can then be copied to the production workspace.

To use this technique, you must mark the State History table **Sharable** in the server group.

Having multiple workspaces run against the same State History table has this limitation: the application does nothing to guard against simultaneous access of the same history record. This means that if two workspaces use the same State History table and process the same set of customer IDs, it is possible for each workspace to access the same history record simultaneously. If two workspaces access the same State History record at the same time, the application does not fail, but some data may not be saved.

If the risk of losing data is small, then you could run two sets of triggers simultaneously for the same set of IDs, one with real time feeds and the other with batch feeds.

The application provides a way to guard against inadvertent access of two workspaces against the same State History table. If a table is not marked **Sharable** in the server group, it can be mapped only on the Deployment tab of a workspace, and it can be used by only one deployment configuration at a time.

Related reference:

"Fields and buttons on the Deployment tab" on page 41

Running workspaces

Use this procedure to run a workspace for test or production purposes.

Before you begin

Only valid workspaces can be run. To be valid, a workspace must contain at least one Simple component and one Action component.

If the workspace was modified after it was deployed, indicated by  next to its name, you must deploy it again for the changes to go into effect when processing transactions.

Procedure

1. Select the workspace you want to run.
   The Workspace Manager page opens.
2. On the Batch Run panel, select the deployment configuration you want to run.
   Only configurations that have been deployed are available for selection.
3. Complete run parameters as desired.
4. Click **Start**.

Related concepts:

Chapter 6, "Outcome data in Opportunity Detect," on page 47

Related tasks:

"Creating and deploying a deployment configuration" on page 40

Fields and buttons on the Batch Run tab

On the Batch Run tab, you can manage run parameters, run deployed workspaces, see the status of the most recent batch runs, and stop and start ongoing runs.

Table 14. Fields and buttons on the Batch Run tab

Field or button	Description
Deployment Configuration	Select a deployment configuration to run and click **Start**. Only deployment configurations that are successfully deployed and ready to run are listed. If you expect a deployment configuration to be available and it is not listed, check the Streams server logs and correct any problems. Then, start the deployment configuration and try to perform the run again. If any transaction data source in the deployment configuration has been mapped to the Web Service connector, the deployment configuration is not available for batch runs.
Run Parameters	
Feed Path	Location of the feed files.
Inactivity Mode	If your workspace includes any Forward Inactivity components or Pattern components configured with the negative event mode, to enable the component to process data as expected you must select **On** in this field and also set the date in the **Inactivity Date** field.
Inactivity Date	Specifies the date the system uses to check for inactivity. Applies only to workspaces that contain Forward Inactivity components or Pattern components configured with the negative event mode, when the **Inactivity Mode** field is set to **On**. It is possible that no transactions for a customer whose transaction has activated the component will be processed when the component is due to fire. In those cases, the system uses the date set in the Inactivity Date field to determine whether the component should fire. Typically, you should set this field to the latest transaction timestamp in the batch file being processed.

Table 14. Fields and buttons on the Batch Run tab (continued)

Field or button	Description
Artificial Transaction	Check this box if your workspace includes any components that use an artificial transaction. You can select the following options. • Off - No artificial transaction is set. • End of Day - Sends an artificial transaction to indicate the end of day at the end of each unique transaction date in the feed files. • End of Run - Sends an artificial transaction after all transactions are sent for a user, regardless of how many transaction dates are in the feed files.
System Log Level	Set the logging level for a batch run. Optionally, select a system log level for real time runs. Options are as follows, listed in ascending order of detail. • Fatal • Error • Warning • Information • Trace • Debug Raising the log level can affect performance. You can also update the logging level during runs by selecting a deployment and clicking  **Update Logging level** on the Deployment tab.
Run in Recovery	When this box is selected, if a run fails, and you re-start the run after fixing any problems, the system attempts to resume the run where it left off. You must wait three minutes for the automatic save to complete before attempting to re-start the run.
Batch Notification	Select this box if you have a batch notification file and you want to receive notifications.
Most Recent	
 Stop button	Click to stop a batch run.
 Start button	Click to re-start a batch run that has been stopped.
Run id	Unique identifier for the most recent run.
Detail id	A counter that increments each time a recovery run occurs.
Deployment Configuration	The name of the deployment configuration used for the run.
Start Time	Date and time of the beginning of the run.
End Time	Date and time of the end of the run.
Run Time	Duration of the run.

Table 14. Fields and buttons on the Batch Run tab (continued)

Field or button	Description
State	During the run, this column may display a variety of interim states. When a run ends, the state is one of the following. • Run Success • Run Failed

Related reference:

"Fields and buttons on the Deployment tab" on page 41

"Fields on the Batch History tab" on page 44

The profile data refresh mechanism

Opportunity Detect automatically refreshes profile data every six hours. If you require an immediate refresh of profile data, you can stop and restart your deployment.

Creating and deploying a deployment configuration

Use this procedure to create and deploy a deployment configuration.

Procedure

1. Select the valid workspace for which you want to create a deployment configuration and click  **Add a Deployment Configuration** on the Deployment tab.

   A panel opens with three tabs: Properties, Data Source Mapping, and History.
2. Complete the fields on the Properties tab and select a server group.

   The type of data source connector that you choose for the **Input mode** field determines what connector types are valid for the data sources in your workspace.
3. If you want to override the default data source mapping for the server group, do the following.
   a. On the Data Source Mapping tab, select the data source you want to change.
   b. Clear the Server Group Default checkbox.
   c. Select an alternate mapping for the data source and click **OK**.

      Note the following restrictions.

      All transaction (input) data sources must be mapped to same connector type.

      The system enforces restrictions on what combinations of data source connectors you can use for transaction, profile, Outcome, and State History data sources.

   These changes apply only within the deployment configuration; they do not affect the default mappings for the server group.
4. Click  **Save** and then click  **Deploy**.

What to do next

When the deployment configuration is successfully deployed, you can use it to run your workspace.

Related tasks:

"Running workspaces" on page 37

Fields and buttons on the Deployment tab

On the Deployment tab of a workspace, you can create deployment configurations for workspaces, map data source connectors within the deployment configuration, deploy the configurations, stop and restart deployments, and view deployment history.

Table 15. Fields and buttons on the Deployment tab

Field or button	Description
 Add button	Click to open a panel where you can create a deployment configuration.
 Save button	Click to save a deployment configuration.
 Delete button	Click to delete the selected deployment configuration.
 Deploy button	Click to deploy the selected deployment configuration.
 Start button	Click to re-start a deployment that has been stopped.
 Stop button	Click to stop a running deployment.
 Update logging level	Click to update the logging level during a run. Raising the log level can affect performance.
Status	Indicates whether all data sources are mapped for the deployment configuration. You can not deploy an incomplete deployment configuration.
Deployment Configuration	Names of the available deployment configurations. Select the radio button next to a name to select the deployment configuration, after which you can deploy it, run it, or modify its properties in the panel that opens.
Server Group	Name of the server group the selected deployment uses.
Deployed	Indicates whether the selected configuration has been deployed.

Table 15. Fields and buttons on the Deployment tab (continued)

Field or button	Description
Version	For batch runs, this number shows the version number of the most recent successful deployment. When Opportunity Detect is integrated with Interact, the number shows the version most recently deployed, re-deployed, or undeployed from IBM Interact.
Input mode	The input mode selected on the Properties tab of the deployment configuration.
Input version	The input version entered on the Properties tab of the deployment configuration.
Output version	The output version entered on the Properties tab of the deployment configuration.
Workspace modified on	Timestamp of the most recent workspace modification. By comparing this timestamp to the timestamp in the **Deployed On** column, you can determine whether the latest version of the workspace is deployed.
Deployed On	Timestamp of the most recent successful deployment of the selected deployment configuration.
Properties tab	
Name	Enter a name for the deployment configuration. Deployment configuration names must be unique across all workspaces.
Server Group	Select from a list of available server groups.
Usage	The usage entered in the definition of the selected server group.
Input mode	Select the mode that matches the type of data source connector you are using for transaction data sources in the deployment. The option selected in this field governs the validation rules applied on Deployment tab.
Input file directory	Applies only when **Input mode** is **Real time file**. The directory in which real time files are placed for processing. In a distributed environment, the input and processed directories must be on the Runtime server and Streamsadmin user Read and Write permission.
Processed file directory	Applies only when **Input mode** is **Real time file**. The directory in which real time files are placed after they are processed.
Input version	If this is the first time you are deploying, enter 1. Increment this number each time you make any changes in the structure of your transaction data and redeploy. If you are using queue connectors, this number must also be updated in the sending program for transaction data.
Output version	If this is the first time you are deploying, enter 1. Increment this number each time you make any changes in the structure of your Outcome data and redeploy.
Script file name (Optional)	If you use a notification batch file, enter the file name here.
Data Source Mapping tab	
Status	Indicates whether a data source is mapped to a connector.

Table 15. Fields and buttons on the Deployment tab (continued)

Field or button	Description
Data Source	Data sources used in the workspace are listed here. Click a link to view the Data Source Connector Mapping dialog box. You can retain the default mapping used in the server group selected on the Properties tab, or you can deselect the **Server Group Default** check box to change the mapping.
Type	Lists the configured data source type: Transaction, State, or Outcome.
Connector	Lists the connector used with the data source for this deployment configuration.
Connector Type	Lists the connector type of the data source: Table, Batch file, Real time file, TCP, Web Service Connector, Queue, or Expanded Outcome. • The Table connector is used for State History tables, for profile data stored in database tables, and for Outcome data where the XML format used by earlier versions of Opportunity Detect is desired. • The Batch file connector is used for profile and transaction data in flat file format. • The Real time file connector is used for transaction data in fixed width file format. • The TCP connector is used only for Interact Advanced Patterns, which is integrated with Interact. • The Web Service connector can be used for transaction data and Outcomes. Special configuration is needed to use the Web Service connector. For more information, see the *IBM Opportunity Detect Administrator's Guide*. • The Queue connector can be used for transaction data and Outcomes. Special configuration is needed to use the Queue connector. For more information, see the *IBM Opportunity Detect Administrator's Guide*. • The Expanded Outcome connector is used for Outcome data sources when the output needs to be stored in database tables structured differently from the format used in the Table connector. Use this connector for integration with IBM Campaign.
Database	Lists the database configured for the data sources that use a Table connector.
Default	Indicates whether the data source uses the default mappings for the server group selected on the Properties tab.
Data Source Connector Mapping window	
Data Source Name	Name of the selected data source.
Connector Type	Available connector types. The connector types that are available are determined by the type of the data source and the input mode.
Table Type	For Outcome data sources, when the connector type is **Table**, you can select either **Outcome** or **Expanded Outcome** for the table type.
Connector	Data source connectors available for mapping.
Database Connection	For Table type connectors, the database connections available for mapping.

Table 15. Fields and buttons on the Deployment tab (continued)

Field or button	Description
Server Group Default	Indicates whether the mapping is the one defined in the server group. Deselect this box if you want to change the data source connector mapping.
History tab	
From Date, To Date, Get History button	Select a date range and click **Get History** to see details for the successful deployments of the selected deployment configuration.
Status	For batch mode, indicates whether the version of the selected deployment configuration was successfully deployed. When Opportunity Detect is integrated with Interact, Undeploy and Redeploy are additional actions for which success or failure can be indicated.
Version	The number of the deployment for which the row provides details. Includes successful and failed deployments.
Action	For stand-alone Opportunity Detect, the only action is Deploy. When Opportunity Detect is integrated with Interact, Undeploy and Redeploy are additional actions.
Date	Timestamp of completion of the listed action.
Input mode	The input mode selected on the Properties tab of the deployment configuration.
Input version	The input version entered on the Properties tab of the deployment configuration.
Output version	The output version entered on the Properties tab of the deployment configuration.
Message	Additional details for the listed action, including the deployment configuration ID, which is automatically generated when deployment takes place.

Related concepts:

"About data source connectors" on page 28

"Data source connector mapping" on page 36

Related reference:

"Fields and buttons on the Batch Run tab" on page 38

"Fields on the Batch History tab"

Fields on the Batch History tab

On the Batch History tab, you can see details about past runs, filtered by deployment and a date range.

Table 16. Fields on the Batch History tab

Field	Description
Deployment Configuration	Select a deployment configuration for which you want to retrieve run history.

Table 16. Fields on the Batch History tab (continued)

Field	Description
From Date, To Date, Get Run History button	Enter or select a date range and then click **Get Run History** to retrieve information about past runs.
Result	The final status of the run. • Run Success • Run Success with Errors • Run Failed
Run Id	Unique identifier for the run.
Detail Id	A counter that increments each time a recovery run occurs.
Run Type	Indicates whether the run is a normal or recovery run.
Start Time	Date and time of the beginning of the run.
End Time	Date and time of the end of the run.
Run Time	The duration of the run.

Related reference:

"Fields and buttons on the Deployment tab" on page 41

"Fields and buttons on the Batch Run tab" on page 38

About batch notifications

The Opportunity Detect engine can run batch files when it completes a run. You can use these files to issue different notifications for each successful and failed engine run.

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

The batch file must be a shell script with the `.sh` extension. It must have executable permission (755).

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

Batch file usage examples

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

Batch file required location

You must create a `scripts` directory on the machine where Streams is installed, under the `/home/streamsadmin/OpDetection` directory, if it has not been created previously. Place your batch file in this directory.

Related concepts:

"Outcome format with the Expanded Outcome data source connector" on page 49

Chapter 6. Outcome data in Opportunity Detect

Opportunity Detect generates one outcome record for every set of transactions that meets the criteria specified in the trigger system.

Outcome data can have different formats, depending on the data source connector that you use for your Outcomes when you run your trigger systems.

You assign data source connectors to tables, queues, or a web service when you create server groups. When you deploy a trigger system, you can change this default mapping in the deployment configuration.

How Outcome data is specified in trigger systems

Outcome data is specified in the Action component, in the Outcome panel. This panel has two fields:

- **Message**, which holds a text string that you specify.
- **Additional information**, which holds data captured using an inline expression. The inline expression can choose data from any data source or component that is available to the Action component.

  This data can be a single value (scalar), or a row in a data record stored in a Container or Select component (tabular).

Types of data source connectors for Outcomes

For Outcome data, Opportunity Detect provides four types of data source connectors.

Outcome table
: Used when you want Outcome data to be stored in database tables in the XML format used by previous versions of Opportunity Detect.

Expanded Outcome tables
: Used when you want Outcome data to be stored in database tables in a form that external systems or IBM Campaign can use more easily than the format provided by the Outcome connector.

Web Service
: Used when your system is configured to use the Opportunity Detect web servlet for Outcome data, for real time operation. See the *IBM Opportunity Detect Administrator's Guide* for details about the Java classes that must be developed to use the Web Service data source connector.

Queue
: Used when your system is configured to use a Queue connector for Outcome data, for real time operation. See the *IBM Opportunity Detect Administrator's Guide* for information on setting up a Queue data source connector.

Related concepts:

"About data source connectors" on page 28

Related tasks:

"Running workspaces" on page 37

Outcome format with the Outcome data source connector

When the Outcome data source connector is used for Outcome data, the system writes the data in the XML format used by previous versions of Opportunity Detect.

Fields in the Outcome table

The following table describes the fields in the Outcome table when you use the Outcome data source connector.

Table 17. Fields in the Outcome table

Field	Description
AUDIENCEID	ID of the audience for which the trigger system fired. Examples of an audience are account, customer, or household.
RUNID	ID of the run. The Run ID helps distinguish between the Outcomes of one run versus the Outcomes of runs before or after it. Because of the Run ID, you do not need to truncate the Outcome table after every run because you can query the table for all of the Outcomes in a specific run.
COMPONENTID	Unique ID of the Action component that fired to generate the Outcome.
AUDIENCETYPE	The single character audience code assigned on the Opportunity Detect Audience Levels page.
OUTCOMEDATE	The timestamp of the final event that caused the Action to fire.
MESSAGE	The data that was specified in the **Message** and **Additional Information** fields of the Action component. The data is written in XML format.
ROW_NUMBER	A unique sequence field, generated automatically.

XML format of the `message` field of the Outcome table

The XML in the `message` field of the Outcome table contains the data specified in the **Message** and **Additional Information** fields of the Action component.

Here is an example of the XML for a simple Outcome. The Action component in the example includes the following information in the Outcome.

- **Message** field: `Send printer paper offer.`
- **Additional Information** field:
  - The sum of all values in the field named `Amount` in a Container component named `All Transactions`.
  - The field named `zipcode` in the data source named `Customer Profile`.

```
<OUTPUT>
 <TEXT>
  Send printer paper offer.
 </TEXT>
 <CONTAINER name"All Transactions" field="Amount" function="SUM">
  123.45
 </CONTAINER>
```

```
 <DATASOURCE name="Customer Profile" field="zipcode">
  11746
 </DATASOURCE>
</OUTPUT>
```

XML format of Outcome values from Container and Select components

Select and Container components can hold values that consist of a single value or a set of fields.

If the value has been aggregated into a single number using a function, it is called a scalar value. Here is an example of the XML for a scalar value where the Outcome includes a field named `Amount` from a Container component named `C1`. The `Amount` field is aggregated using the `average` function.

```
<CONTAINER name="C1" field="Amount" function="average">
 123.45
</CONTAINER>
```

If the value is a set of fields, it is called a tabular value. Here is an example of the XML for a tabular value where the Outcome includes fields named `Field_1`, `Field_2`, and `Field_3` from a Select component named `S1`.

```
<SELECT name="S1">
 <ROW>
  <FIELD name="Field_1">abc</FIELD >
  <FIELD name="Field_2">123.45</FIELD >
  <FIELD name="Field_3">xyz</FIELD >
 </ROW >
</SELECT >
```

Outcome format with the Expanded Outcome data source connector

When the Expanded Outcome data source connector is used for Outcome data, the system writes the data in a form that can be used by IBM Campaign or an external system.

The data produced by Opportunity Detect is called Outcome data.

About the Expanded Outcome tables

The Expanded Outcome connector writes the Outcome data to two database tables, which you must create using scripts provided with Opportunity Detect.

DB2 is the only supported database type for the Expanded Outcome tables.

The tables are:

- A **primary** table that contains the text string specified in the **Message** field in the Action component.
- A **secondary** table that contains the data specified in the **Additional information** field in the Action component.

You provide a base name for the Expanded Outcome tables when you run the `ExpandedTable.sql` script to create the tables. The script appends the number 1 to the name of the primary table, and appends the number 2 to the name of the secondary table.

For example, if you specify the base name `ExpandedOutcome`, the script creates two tables: `ExpandedOutcome1` and `ExpandedOutcome2`.

Fields in the Expanded Outcome tables

These descriptions of the fields in the Expanded Outcome tables refer to scalar and tabular values, which are defined as follows:

Scalar

A single unit of data.

Tabular

A data set, as in a database row. In Opportunity Detect Outcomes, tabular data is saved in XML format.

Depending on how you specify the Outcome data, the Outcome can contain either type of value, or both types. If you include tabular data in a Campaign integration, additional processing is required before Campaign can consume it.

Table 18. Fields in the Expanded Outcome primary table

Field	Description	Data type
OUTCOMEID	Unique sequence ID. Used as the primary key to link to the secondary Expanded Outcome table.	Integer
AUDIENCEID	ID of the audience member for which the trigger system fired. Examples of an audience are account, customer, or household. The audience ID is stored as a string. Multi-column audience IDs are not supported.	NVARCHAR(60) If you use Oracle system tables and plan to integrate with Campaign, you must change the data type of this field from NVARCHAR(60) to Varchar2(60) because Campaign does not support the NVARCHAR(60) data type.
AUDIENCELEVEL	The single character audience code assigned on the Opportunity Detect Audience Levels page.	NVARCHAR(60) If you use Oracle system tables and plan to integrate with Campaign, you must change the data type of this field from NVARCHAR(60) to Varchar2(60) because Campaign does not support the NVARCHAR(60) data type.
COMPONENTID	Unique ID of the Action component that fired to generate the Outcome.	Varchar
OUTCOMEDATE	The timestamp of the final event that caused the Action component to fire.	Timestamp
RUNID	ID of the run, for batch mode only. The Run ID helps distinguish between the Outcomes of one run versus the Outcomes of runs before or after it. Because of the Run ID, you do not need to truncate the Outcome table after every run because you can query the table for all of the Outcomes in a specific run.	Integer

Table 18. Fields in the Expanded Outcome primary table (continued)

Field	Description	Data type
MESSAGE	The text string that was specified in the **Message** field of the Action component.	NVARCHAR(60) If you use Oracle system tables and plan to integrate with Campaign, you must change the data type of this field from NVARCHAR(60) to Varchar2(60) because Campaign does not support the NVARCHAR(60) data type.
PROCESSED	A flag that indicates whether the data has been consumed by Campaign.	Integer

Table 19. Fields in the Expanded Outcome secondary table

Field	Description	Data type
OUTCOMEID	Unique sequence ID. Used as a foreign key to link the record to the primary Expanded Outcome table.	Integer
NAME	The name assigned in the **Additional Information** field of the Action component.	NVARCHAR(60) If you use Oracle system tables and plan to integrate with Campaign, you must change the data type of this field from NVARCHAR(60) to Varchar2(60) because Campaign does not support the NVARCHAR(60) data type.
VALUE	The scalar and tabular data that was specified in the **Additional Information** field of the Action component. Tabular values are saved in XML format.	Clob
DATATYPE	For scalar values, the data type can be one of the following. • boolean • currency • date • double • integer • string For tabular values, the data type is set to string, because tabular values are stored in XML, and the data type for XML is string.	NVARCHAR(60) If you use Oracle system tables and plan to integrate with Campaign, you must change the data type of this field from NVARCHAR(60) to Varchar2(60) because Campaign does not support the NVARCHAR(60) data type.

XML format of tabular values

Here is an example of the XML for a tabular value, where the record includes these fields:

- `Field_1`
- `Field_2`
- `Field_3`

Example

```
<SELECT name="S1">
 <ROW>
  <FIELD name="Field_1">abc</FIELD >
  <FIELD name="Field_2">123.45</FIELD >
  <FIELD name="Field_3">xyz</FIELD >
 </ROW >
</SELECT >
```

Integrating Opportunity Detect with Campaign in batch mode

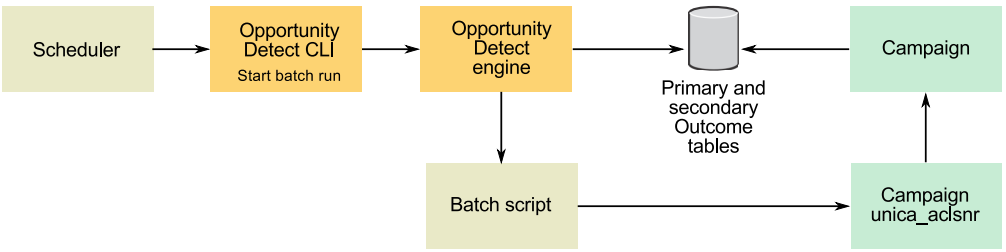
The following example illustrates how you might use the Expanded Outcome data in Campaign, in batch mode.

Before you begin

You must have Campaign and Opportunity Detect installed and running.

About this task

The following diagram illustrates the example described in this procedure.



Procedure

1. Create the Expanded Outcome tables in your database using the script provided with Opportunity Detect.
2. On the Server Groups page in Opportunity Detect, do the following.
   - If a database connection for the database where you created your Expanded Outcome tables does not exist, create one.
   - If an Expanded Outcome data source connector does not exist, create one.

     If you make the connector sharable, you can map the connector to your primary Expanded Outcome table on the Server Groups page or on the Deployment tab of the workspace. If you do not make the connector sharable, you can map it only on the Deployment tab.
3. Create the Opportunity Detect workspace and configure it to use the Expanded Outcome data source connector for Outcome data, either on the Server Groups page or on the Deployment tab of the workspace.
4. On the Deployment tab of the Opportunity Detect workspace, configure the deployment to call a batch file at the end of a successful run.

   Create the batch script to call the Campaign listener service, `unica_aclsnr`, to run a Campaign flowchart that you design.
5. Use the Opportunity Detect command line utility, `RemoteControlCLI` (CLI), to run the workspace.

   Use your own scheduling utility to run the CLI batch script at the desired interval; for example, daily.

   When the workspace runs, Opportunity Detect inserts Outcome data into the Expanded Outcome tables.

6. Configure your Campaign flowchart as follows.
   a. In a Select process, create a new table mapping as follows.
      - Map your main audience in Campaign to the OUTCOMEID field in the primary Expanded Outcome table. This is required so that you can select Outcome records for use in the flowchart. Selection must use the OUTCOMEID field, as the same AUDIENCEID field can be repeated in multiple Outcome records.
      - Map your alternate audience in Campaign to the AUDIENCEID field in the primary Expanded Outcome table. This mapping defines the audience on which rest of the flowchart logic should be performed.

      Note: If you plan to use Opportunity Detect Outcome data in multiple flowcharts, save the mapped table information into a table catalog and load this catalog in other flowcharts.
   b. Select records where the value in the `PROCESSED` field in the primary Expanded Outcome table is 0.

      This value indicates that the record has not been processed yet.
   c. Set the value in the `PROCESSED` field in the primary Expanded Outcome table to 1, to indicate that the record has been processed.

      You can write SQL in a Select process to set this value.
   d. In an Audience process, switch the audience from OUTCOMEID to AUDIENCEID.
   e. Use the Opportunity Detect data as desired in your flowchart.
   f. Use a MailList process to assign an offer and update contact history.

Integrating Opportunity Detect with Campaign in interactive mode

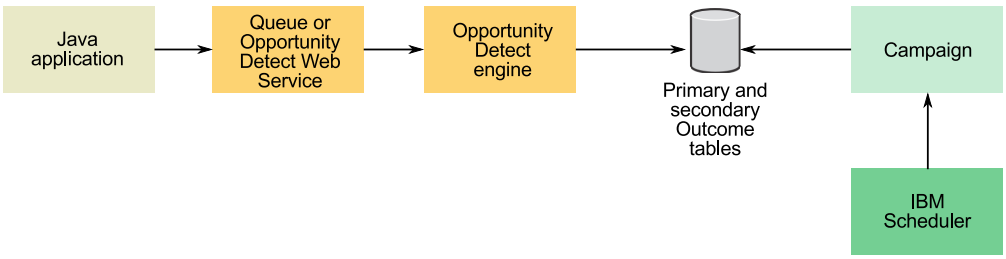
The following example illustrates how you might use the Expanded Outcome data in Campaign, in interactive mode.

Before you begin

You must have Campaign and Opportunity Detect installed and running.

About this task

The following diagram illustrates the example described in this procedure.



Procedure

1. Create the Expanded Outcome tables in your database using the script provided with Opportunity Detect.
2. Do one of the following.

- If you plan to use a queue connector, configure a queue for your transaction data in your queue server.
- If you plan to use the Web Service, develop the required Java™ classes.

3. On the Server Groups page in Opportunity Detect, do the following.
   - If a database connection for the database where you created your Expanded Outcome tables does not exist, create one.
   - If an Expanded Outcome data source connector does not exist, create one.

     If you make the connector sharable, you can map the connector to your primary Expanded Outcome table on the Server Groups page or on the Deployment tab of the workspace. If you do not make the connector sharable, you can map it only on the Deployment tab.
4. Configure the Opportunity Detect workspace to use the Web Service or a queue data source connector for transaction data, and the Expanded Outcome data source connector for Outcome data.
5. Configure your Campaign flowchart as follows.
   a. In a Select process, create a new table mapping as follows.
      - Map your main audience in Campaign to the OUTCOMEID field in the primary Expanded Outcome table. This is required so that you can select Outcome records for use in the flowchart. Selection must use the OUTCOMEID field, as the same AUDIENCEID field can be repeated in multiple Outcome records.
      - Map your alternate audience in Campaign to the AUDIENCEID field in the primary Expanded Outcome table. This mapping defines the audience on which rest of the flowchart logic should be performed.

      Note: If you plan to use Opportunity Detect Outcome data in multiple flowcharts, save the mapped table information into a table catalog and load this catalog in other flowcharts.

   b. Select records where the value in the `PROCESSED` field in the primary Expanded Outcome table is 0.

      This value indicates that the record has not been processed yet.

   c. Set the value in the `PROCESSED` field in the primary Expanded Outcome table to 1, to indicate that the record has been processed.

      You could write SQL in a Select process to set this value.

   d. In an Audience process, switch the audience from OUTCOMEID to AUDIENCEID.
   e. Use the Opportunity Detect data as desired in your flowchart.
   f. Use a MailList process to assign an offer and update contact history.
6. Use your own scheduling utility or the IBM Marketing Software Scheduler to schedule flowchart runs at the desired interval; for example, every minute.

Outcome format with the Web Service data source connector

You can use a Web Service Connector for incoming transaction data and for outcome data.

When you use the Web Service connector for outcomes, the data is sent to a web servlet. This servlet executes a plugin that can write to a file. You can also do additional programming to enable the plugin to write to a queue or act like a web service.

See the *IBM Opportunity Detect Administrator's Guide* for details about the Web Service connector.

Where to find Outcome statistics

You can view statistics about your Outcomes in the Streams logs.

Each engine reports the number of Outcomes sent when the End of Cycle occurs. Look for a message like the following:

```
Sent N outcome messages for O1.
```

where O1 is the name of the outcome datasource in Streams.

These appear as an INFO level log entry.

Draft comment

Would they be able to see anything about Outcomes in the Montoring utility?

Chapter 7. Queue connectors for input and output in Opportunity Detect

The Queue data source connector allows Opportunity Detect to operate with a supported queue server.

A Queue data source connector is a good choice when a high volume of transactions must be processed. Queues handle spikes in demand more efficiently than the Web Service does. Also, unlike the Web Service, queues retain messages in the event of a network or machine failure, or if you deploy a new version of a configuration deployment.

To use the Queue data source connector, your organization must develop code to send the transaction data and consume the Outcome data, and you must install and maintain a queue server.

Opportunity Detect supports both Java Message Service (JMS) and Advanced Message Queuing Protocol (AMQP) queue servers. See the *IBM Marketing Software Recommended Software Environments and Minimum System Requirements* document for a list of supported queue servers.

Related concepts:

Chapter 3, "Real time processing in Opportunity Detect," on page 9

Chapter 8, "Web Service connectors for input and output in Opportunity Detect," on page 69

"About data source connectors" on page 28

Related reference:

"Fields on the Server Groups page" on page 21

Transaction message examples for the sending program

This section describes the data in the message that your sending program must produce when you use a Queue data source connector for transaction data. It also provides code samples for creating a transaction message in the required format.

Information required for the transaction message

When you use a Queue data source connector for transaction data, the sending program must provide the following information in the data sent to the queue server.

For AMQP queues, the message has two parts:

- Properties name value pairs including a custom header
- Data as json payload

Table 20. Required information for transaction data

Name	Value
AUDHASH	Hash for the audienceId generated using a standard hash function. The function must generate a positive value. Used in the header of a JMS message. Not required for AMQP.
deploymentConfigurationId	You can obtain the ID of the deployment configuration on the Deployment tab of the workspace. Select the deployment configuration and select the History tab in the panel that opens. Look under the Message column for the configuration ID. Used in the header and the body of a JMS message. Required in the AMQP message custom header.
inputVersion	The input version number entered on the Properties tab of the deployment configuration. Used in the header and the body of a JMS message. Required in the AMQP message custom header.
AudienceLevel	The code assigned to the audience level on the **Settings > Detect Settings > Audience Levels** page. Used in the header and the body of a JMS message. Required in the AMQP message custom header.
audienceId	The customer ID associated with this transaction. Used in the header and the body of a JMS message. Required in the AMQP message custom header.
inputDataSourceName	The name of the data source to which the Queue data source connector is mapped. Data sources are defined on the **Settings > Detect Settings > Data Sources** page. Used in the header and the body of a JMS message. Required in the AMQP message custom header.

transaction	Fields from your Transaction data source, as defined on the **Settings > Detect Settings > Data Sources** page.
	
	Used in the header and the body of a JMS message. Required in the AMQP message custom header.
	Note: The date must be expressed in milliseconds.
	
	JMS example:
	`{"FLAG":true,`
	`"DATE":1408077610000,`
	`"AMT":1000.00,`
	`"NUMBER":10,`
	`"TRANCODE":"ATMD",`
	`"CURRENCY":1009.1,`
	`"ID":"cm00411"}`

Sample code for creating a transaction message for JMS queue servers

Here is a sample code snippet for creating a JMS message for transaction data using the JMS API.

Line breaks have been added for readability.

```
//JMS Header Fields
  MapMessage message = session.createMapMessage();
  message.setLongProperty("AUDHASH",hashcode);
   message.setStringProperty("deploymentConfigurationId",
        "eecdc1b4-5333-4076-9af6-dc6c2f932c0e");
   message.setLongProperty("inputVersion", 5);

//JMS Body Fields
   message.setString("deploymentConfigurationId",
         "eecdc1b4-5333-4076-9af6-dc6c2f932c0e");
   message.setInt("inputVersion", 5);
   message.setString("audienceLevel", "audienceLevel");
   message.setString("audienceId", "cm00409");
   message.setString("inputDataSourceName", "DataSourceNam");
   message.setString("transaction", "{\"Date\":1408077610000,\"Amount2\":1000.00,\
         "Type\":\"E1a\",\"Amount1\":1009.1,\"ID\":\"cm00409\"}");
```

Sample code for creating a transaction message for AMQP queue servers

The client must ensure reliable message delivery.

Sender programs send transaction to the input exchange specified during deployment by specifying the routing key. This key should be within the number of engines used by the deployment and should remain the same for the audienece ID (that is, it should be sticky).

For example, if a a deployment has 16 engines, then a routing key should be generated with the following characteristics.

- The key remains the same for each audience I.D
- The key should not be greater than 16.
- The key evenly distributes the load among 16 engine.

One way to generate a routing key is to use the modulo operation. For example, if there are 8 engines then for the audience ID hash the routing key can be calculated as follows. The remainder after division by 8 is the routing key.

Table 21. Routing key example

Audience ID hash	Modulo operation	Routing key
9	9 mod 8	1
11	11 mod 8	3
8	8 mod 8	0 The routing key starts with 0.

Line breaks have been added for readability.

```
int routekey = audience % engineCount+1;
BasicProperties.Builder propsBuilder = new BasicProperties.Builder();
Map<String, Object> headers = new HashMap<String, Object>();
headers.put("deploymentConfigurationId", deploymentConfigurationId);
headers.put("inputVersion", inputVersion);
headers.put("audienceLevel", "NOTREQD");
headers.put("audienceId", audienceId);
headers.put("inputDataSourceName", inputDataSourceName);
propsBuilder.headers(headers);
channel.basicPublish("InputExchage", routekey+"", propsBuilder.build(),
   dataString.getBytes("UTF-8"));
```

Example of a transaction message

This is sample input message that was sent to an Active MQ queue server.

Line breaks have been added for readability.

```
ActiveMQMapMessage {commandId = 0, responseRequired = false,
messageId = ID:ADMINIB-3C6H892-58046-1428399823708-0:0:1:1:1,
originalDestination = null,
originalTransactionId = null,
producerId = null,
destination = queue://amqinput,
transactionId = null,
expiration = 0,
timestamp = 1428399824137,
arrival = 0,
brokerInTime = 0,
brokerOutTime = 0,
correlationId = null,
replyTo = null,
persistent = false,
type = null,
priority = 4,
groupID = null,
groupSequence = 0,
targetConsumerId = null,
compressed = false,
userID = null,
content = org.apache.activemq.util.ByteSequence@60945fea,
marshalledProperties = null,
dataStructure = null,
redeliveryCounter = 0,
size = 0,
properties = {inputVersion=5,
     deploymentConfigurationId=eecdc1b4-5333-4076-9af6-dc6c2f932c0e,
     AUDHASH=13670},
     readOnlyProperties = false,
     readOnlyBody = false, droppable = false}
ActiveMQMapMessage
{ theTable =
     {audienceLevel=audienceLevel,
     audienceId=cm00409,
     inputVersion=5,
     transaction={"FLAG":true,
        "DATE":1408077610000,
        "AMT":1000.00,
        "NUMBER":10,
        "TRANCODE":"ATMD",
        "CURRENCY":1009.1,
        "ID":"cm00411"}
 }
```

Outcome message examples

This section describes the data in the message that the system produces when you use a Queue data source connector for Outcomes. It also provides code samples for consuming an Outcome message.

Outcome message format using the Queue data source connector

When you use a Queue data source connector for the Outcome data source the message has the format shown in the following table.

In addition, the following rules apply.

- System-defined field names are camel case.
- Data types start with a capital letter.
- Valid data types are String, Integer, Double, Currency, Boolean, Date, and Tabular.
- Values for Date and String data types are surrounded with double quotes.
- Values for Integer, Double, Currency, Boolean, and Tabular data types are **not** surrounded with double quotes.

For AMQP, all the fields below except data are part of the AMQP custom header. Data is the payload part of the message.

Table 22. Message format for Outcome data

Name	Value
producerName	OpportunityDetect
clientID	If you have multiple Opportunity Detect installations, enter the ID of the installation where this Queue data source connector is used, as defined in the **clientId** property under the **IBM Opportunity Detect And Interact Adv Patterns | System** category on the **Settings > Configuration** page. If you have a single Opportunity Detect installation, this value should be `default`.
actionName	The name of the Action component that produced the Outcome.
workspaceName	The name of the workspace.
deploymentName	The name of the deployment that is running the workspace.
outputVersion	The output version number entered on the Properties tab of the deployment configuration.
data	The data you have specified in the Action component for your Outcome, plus the data that is included in every Outcome. Outcome data always includes the audience ID, audience level, the ID of the Action component, and a time-stamp. Time-stamps have this format: `yyyy-MM-dd HH:mm:ss` Additional fields that you specify in the Action component are output with the label you give the field in the Action component, the value, and the data type. This data can be a single value (scalar), or a row in a data record stored in a Container or Select component (tabular).

Sample code for consuming the Outcome message from JMS queue servers

This section applies to JMS servers. The way you use the Outcome message can vary depending on your needs. The following code snippet is a basic example.

Line breaks have been added for readability.

```
@Override
   public void listenforOutput() throws QueueConnectorException {
      try {
            Message message = consumer.receive(timeout);
            if (message != null) {
                        formatOutputMessage((MapMessage) message);
            } else {
                        System.out.println("No More messages on Queue!\n");
            }
       } catch (JMSException jmsex) {
            recordFailure(jmsex);
            destroy();
       }
 return;
    }
```

Sample code for consuming the Outcome message from AMQP queue servers

The way you use the Outcome message can vary depending on your needs. The following code snippet is a basic example.

Line breaks have been added for readability.

```
String queueName = channel.queueDeclare().getQueue();
channel.queueBind(queueName, queueConfiguration.getDestinationName(), "");
   Consumer consumer = new DefaultConsumer(channel) {
      @Override
      public void handleDelivery(String consumerTag, Envelope envelope,
      AMQP.BasicProperties properties, byte[] body) throws IOException
      {
         String message = new String(body, "UTF-8");
         System.out.println("Received :\n " + ": " + message + "\nHeader :
         \n"+properties);
      }
   };
   channel.basicConsume(queueName, true, consumer);
```

Example of the Outcome message from a JMS queue server

Here is an example of a message produced by the system when you use a Queue data source connector for an Outcome.

Line breaks have been added for readability.

```
producerName: OpportunityDetect,
clientID :default,
actionName:  "Name of Action component"
workspaceName :  "Name of workspace"
deploymentName : "Name of deployment"
outputVersion: "Version",

data = [
  {
    "message": {
        "value": "Pass",
        "dataType": "String"
        }
  },
  {
    "audienceId": {
        "value": "acc333",
        "dataType": "String"
        }
  },
```

```
  {
    "audienceLevel": {
        "value": "audienceLevelvalue",
        "dataType": "String"
        }
  },
  {
    "componentId": {
        "value": "AA123344555666",
        "dataType": "String"
        }
  },
  {
    "firingtime": {
       "value": "2014-04-01 05:40:01",
       "dataType": "Date"
        }
  },
  {
    "fieldName1": {
        "value": 333,
        "dataType": "Integer"
        }
  },
  {
    "fieldName2": {
        "value": "2014-04-01 05:40:01",
        "dataType": "Date"
        }
  },
  {
    "fieldName3": {
        "value": {
            "header" : [
                {
                    "tablefield1": {
                     "dataType": "String"
                    },
                }
                {
                    "tablefield2": {
                     "dataType": "Date"
                    }
                },
            ]
            rows : [
                    {
                      "values" : ["value1","value2"]
                    },
                    {
                      "values" : ["value1","value2"]
                    },
                    {
                      "values" : ["value1","value2"]
                    },
               ]
         },
         "dataType": "Tabular"
      }
   }
]
```

Example of the Outcome message from an AMQP queue server

Output is sent to the queue server as durable message.

```
[ { "componentId" :
    { "value" : "457e7bba-ae6a-4776-828e-f2d4ba174a06" , "dataType" : "String" }
  },
  { "audienceId" :
    { "value" : "Trans_ID" , "dataType" : "String" }
  },
  { "audienceLevel" :
    { "value" : "f2c9770b-44b0-46d1-b068-c9e9dcffbc60" ,
       "dataType" : "String" }
    },
  { "firingtime" :
    { "value" : "2014-08-15 04:40:10" , "dataType" : "Date" }
  },
  { "message" :
    { "value" : "This is outcome" , "dataType" : "String" }
  },
  { "l1" :
    { "value" : 100 , "dataType" : "Integer" }
  } ,
  { "l2" :
    { "value" : "Trans_ID" , "dataType" : "String" }
  } ,
  { "Number" :
    { "value" : 0 , "dataType" : "Integer" }
  } ,
  { "string" :
    { "value" : "" , "dataType" : "String" }
  } ,
  { "double" :
    { "value" : 0 , "dataType" : "Double" }
} ]
```

Queue processing setup roadmap

This roadmap describes how to set up real time processing using queue data source connectors in Opportunity Detect.

Table 23. Real time processing setup roadmap

Step	Where to find details
Ensure that you have met the prerequisites.	"Prerequisites for using queue connectors"
Decide how many queue data source connectors to create and how you want to map them.	"Planning your queue connectors" on page 65
If you are using a JMS queue server, copy client libraries from your queue server to your IBM InfoSphere Streams machine and set environment variables.	"Installing MQ client libraries" on page 66
Set up data sources in Opportunity Detect for the transaction data that you want to process in real time mode.	Chapter 4, "Data source setup in Opportunity Detect," on page 11
Create queue data source connectors for the transaction and Outcome data sources that you want to process in real time mode.	"Creating queue connectors" on page 66
Map your queue data source connectors to your transaction and Outcome data sources.	"Mapping queue connectors to data sources" on page 67

Prerequisites for using queue connectors

Ensure that the following prerequisites are met when you use queue connectors.

1. You must have a functioning supported queue server with queues configured as required for your deployment configuration.
2. You must set up one queue for each queue data source connector you plan to use.
3. The queue server must be accessible from the Opportunity Detect run-time server (the Streams server).
4. The Streams system user, typically the streamsadmin user, must have read and write permissions on the queue server.
5. Your organization must develop the necessary sending program to push transaction data into the queue.
6. If you have multiple Opportunity Detect installations, set an ID for each one in the **clientId** property under the **IBM Opportunity Detect And Interact Advanced Patterns | System** category on the **Settings > Configuration** page. If you have a single Opportunity Detect installation, you do not have to set any values for this property.

Planning your queue connectors

Consider the following points when you create and map queue data source connectors.

- Opportunity Detect supports SSL communication only with the WebSphere MQ queue server. If you are using another type of queue server, your queue server must not be configured to use SSL.
- Ensure that at least one queue is created on your queue server for each queue connector that you plan to use.
- A workspace has only one Outcome data source, regardless of how many Action components it contains.
- You must map one queue connector to each transaction and Outcome data source. Optionally, you can map more than one queue connector to the transaction and Outcome data sources.
- A non-shared queue connector can be used by only one deployment. The system enforces this constraint.
- A non-shared connector cannot be mapped to a data source when you configure a server group. It can be mapped in a deployment configuration only.
- A shared queue connector can be used across multiple deployment configurations.
- Shared connectors are mapped when you configure a server group. You can change this mapping in a deployment configuration.
- For Outcomes, you would typically use multiple non-shared queues if you have multiple consumers of the Outcomes, and use a shared queue if you have only a single consumer of the Outcomes.
- For testing, it is typical to use shared data source connectors for your transaction data, to reduce the time required to set up separate queues. Because the message includes the configuration ID, the transaction data is sent to the appropriate queue. For production, you should generally use non-shared connectors for your transaction data.

Installing MQ client libraries

To use the queue data source connector with a JMS queue server, you must install the message queue client and set the library path in an environment variable on your Streams server. This procedure is not necessary if you use an AMQP queue server.

Procedure

1. Obtain the client for your message queue server.
   - For WebSphere, download the client here:
     http://www-01.ibm.com/software/integration/wmq/clients/
   - For Active MQ, download the client here:
     http://activemq.apache.org/download.html
2. Install the message queue client one every machine where IBM InfoSphere Streams is installed.
   See the documentation for your queue server for full instructions on installing the client.
3. On all machines where Streams is running, edit the Streams `.bashrc` file for the streamsadmin user as follows.
   The `.bashrc` file is typically located in home directory for streamsadmin user.
   a. Set the `JAVA_HOME` environment variable to use Streams Packaged Java. For example:
      ```
      export JAVA_HOME=/home/streamsadmin/InfoSphereStreams/java
      ```
   b. Set additional environment variables, as follows.
      - WebSphere MQ
        Set the `STREAMS_MESSAGING_WMQ_HOME` environment variable to the path where you installed the client library. For example:
        ```
        export STREAMS_MESSAGING_WMQ_HOME="/opt/mqm"
        ```
      - ActiveMQ
        Set the `STREAMS_MESSAGING_AMQ_HOME` environment variable to the location where Apache ActiveMQ is installed. For example:
        ```
        export STREAMS_MESSAGING_AMQ_HOME="/home/streamsuser/
        ApacheActiveMQ"
        ```
4. Restart the Streams instance.

Creating queue connectors

Follow this procedure to create a queue connector.

Before you begin

You must have transaction data sources defined on the **Settings > Detect Settings > Data Sources** page.

Procedure

1. Navigate to the **Settings > Detect Settings > Server Groups** page.
2. On the Data Source Connectors tab, click **Add a new Data Source Connector** and select **Queue**.

3. Complete the fields in the panel that opens. Note that you can make WebSphere MQ and Active MQ queue connectors sharable, but you can not make Rabbit MQ connectors sharable, because with Rabbit MQ the input is consumed in a round robin pattern.

Mapping queue connectors to data sources

Follow this procedure to map a queue connector to a data source in a deployment configuration.

Before you begin

- You must have transaction data sources defined on the **Settings > Detect Settings > Data Sources** page.
- You must have queue connectors defined on the Data Source Connectors tab on the **Settings > Detect Settings > Server Groups** page.

About this task

This procedure assumes that you are mapping the transaction and Outcome data sources to queue connectors in a deployment configuration, overriding the default server group settings.

Procedure

1. Create a deployment configuration as described elsewhere in this guide.
2. On the Data Source Mapping tab, select the transaction data source you want to change.
3. Clear the **Server Group Default** checkbox.
4. In the Data Source Connector Mapping window, do the following.
   a. Select **Queue** as the **Connector Type**
   b. Select a queue connector.
5. Repeat steps 2-4, selecting any additional transaction and Outcome data sources you want to map.

Chapter 8. Web Service connectors for input and output in Opportunity Detect

You can use a Web Service Connector for incoming transaction data, for Outcome data, or for both.

To use the Web Service Connector for either input or Outcome data, your organization must develop Java classes.

- When you specify the Web Service Connector for incoming transaction data, you develop a Java class that sends the data to an API over HTTP.
- When you specify the Web Service Connector for Outcome data, you develop a Java class that implements an interface provided by a servlet. You can write the Outcome data to a file, to a database, or you can develop your plugin to act like a web service.

This section provides the details required for this development effort.

Example use case

Here is an example of how a telephone company might use the Web Service connector for both input and Outcome data.

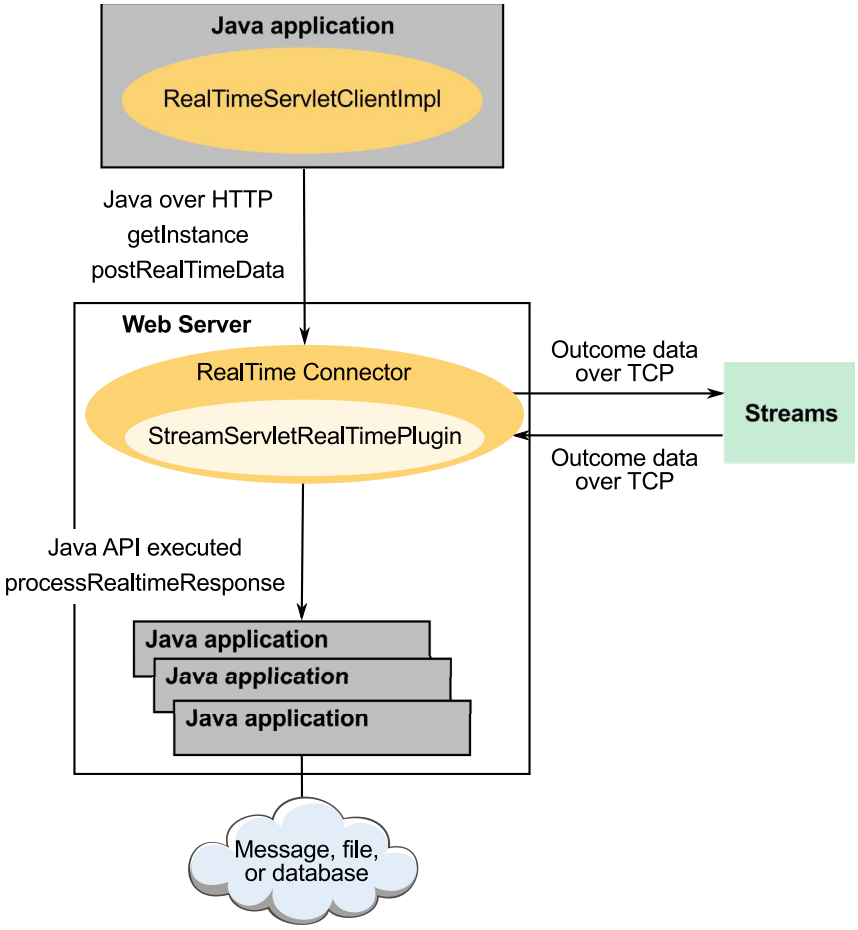
- A customer makes a call on her cell phone while on vacation.
- A call data record is sent to the Web Service connector.
- Opportunity Detect processes the transaction and detects an unusual event, in this case a call from outside the customer's data plan area.
- Opportunity Detect sends the Outcome data to two Web Service plugins.
  - One plugin updates profile variables used by Interact.

    This data improves the customer profile that Interact uses to evaluate effective offers when the customer visits the company's web site or calls customer service.
  - Another plugin alerts a company system, which sends an SMS offer to the customer's phone.

    For example, the message might offer to upgrade her data plan.

Block diagram

The following diagram illustrates the Web Service connector.



Location of the input jar files and the output class and plugin

All of the locations described in this section are under the run time installation of Opportunity Detect.

- For input, your Java application must use the `RealTimeClient.jar` file, located here:

  `RealTimeConnector/wlp/usr/servers/RealTimeConnector/dropins/RealTimeConnector/WEB-INF/client`

  The following dependent .jar files must be included in the Java class path of your application. They are in the same location as the `RealTimeClient.jar` file.

  - `JSON4J.jar`
  - `log4j.jar`

- For Outcomes, your Java application must use the `StreamServletRealTimePlugin` class, located here:

  `RealTimeConnector/wlp/usr/servers/RealTimeConnector/dropins/RealTimeConnector/WEB-INF/classes/com/ibm/unica/detect/interact/servlet`

  Place the plugin you develop here:

  `RealTimeConnector/wlp/usr/servers/RealTimeConnector/dropins/RealTimeConnector/WEB-INF/lib`

Timestamp format

Timestamps in input data must be expressed in milliseconds.

Data types

No objects are used for either input or Outcomes using the Web Service connector. Only primitive data types are used.

Locale

Locale is handled by the Opportunity Detect engine.

Monitoring

Because Opportunity Detect is built on Streams, you can use the built-in monitoring capabilities of the Streams server to monitor processing. See the Streams documentation for details.

Related concepts:

Chapter 3, "Real time processing in Opportunity Detect," on page 9

"About data source connectors" on page 28

"About deployment configurations" on page 35

Chapter 7, "Queue connectors for input and output in Opportunity Detect," on page 57

Chapter 8, "Web Service connectors for input and output in Opportunity Detect," on page 69

Related reference:

"Fields on the Server Groups page" on page 21

Class for input: RealTimeClient

Reference for the `RealTimeClient` class, used for sending input data to the Web Service connector.

You must compile your implementation against `RealTimeClient.jar`.

The `RealTimeClient` class has two methods:

- `getInstance`
- `postRealTimeData`

Code example

Here is a code example. The code opens an HTTP connection to the Interact Connector servlet and sends transaction data.

```
private static
 String
  urlString ="http://detectvm2.emmlabs.ibm.com:8282/servlets/StreamServlet";
 RealTimeServletClient
  streamServletRealTimeClient = RealTimeServletClientImpl.getInstance(urlString);
 String
  deploymentConfigurationId = "e0692797-70fc-4a34-9232-9f7df3e9bb68";
 int version = 5;
 String audienceId = "Acc09";
```

```
String inputDataSourceName = "RealTimeDataSource";
Map inputDataSourceFields = new Hashtable();
 // The names of the keys below are the names of datasource fields.
 // Start of transaction
 inputDataSourceFields.put("Boolean", true);
 inputDataSourceFields.put("Timestamp", new Date().getTime());
 inputDataSourceFields.put("Double", 99.9);
 inputDataSourceFields.put("Integer", 56);
 inputDataSourceFields.put("String", "Hello");
 inputDataSourceFields.put("AudienceId", audienceId);
 // End of transaction
streamServletRealTimeClient.postRealTimeData
 (deploymentConfigurationId, version, audienceId,
  inputDataSourceName, inputDataSourceFields);
```

In the example, the fields in the transaction hash table are defined on the **Detect Settings > Data Sources** page as shown below.

Fields

Name	Display Name	Description	App Type	Data Type	Name Value List
AudienceId	AudienceId		Audience Id : Cust	String	
Timestamp	Timestamp		Transaction Date	Date	
Integer	Integer		Transaction Attrib	Integer	
Double	Double		Transaction Attrib	Double	
String	String		Transaction Attrib	String	
Boolean	Boolean		Transaction Attrib	Boolean	

Additional details are provided in the remainder of this section.

getInstance

```
getInstance (urlString, connectTimeout, readTimeout)
```

urlString

URL of the Interact connector. The value is the same as used for the **Opportunity Detection | System | Real Time Connector | ServerURL** configuration property, shown on the **Settings > Configuration** page.

postRealTimeData

```
postRealTimeData (deploymentConfigurationId, version, audienceId,
inputDataSourceName, inputDataSourceFields)
```

deploymentConfigurationId

The deployment configuration ID. For example `e0692797-70fc-4a34-9232-9f7df3e9bb68`.

To obtain the version number of a deployment configuration, select the Deployment tab of a workspace, select the relevant deployment configuration, and select the History tab in the panel that opens. Use the configuration ID listed in the **Message** column.

audienceId

The value in the audience ID field in your customer data source.

This field must be defined as an audience field on the Data Sources page in Opportunity Detect.

version

The version of the deployment.

To obtain the version number of the deployment, select the Deployment tab of the workspace, select the relevant deployment configuration, and select the History tab in the panel that opens. Use the most recent version listed in the Version column.

If you re-deploy the deployment configuration that you are using, you must update this version parameter in your Java program.

inputDataSourceName

The name of the data source that has been created on the Data Sources page in Opportunity Detect for real time transactions.

inputDataSourceFields

A map of fields in the data source as defined on the Data Sources page in Opportunity Detect.

Class for Outcomes: StreamServletRealTimePlugin

Reference for the `StreamServletRealTimePlugin` class, used for sending Outcome data to the Web Service connector.

Editing the Streams servlet web.xml

To make a plugin executable, you must edit the Streams servlet `web.xml` file.

The Streams servlet `web.xml` file is located under your Opportunity Detect run time installation in the following path:

`RealTimeConnector/wlp/usr/servers/RealTimeConnector/dropins/RealTimeConnector/WEB-INF`.

Add the StreamServletRealTimePlugin class name to the following section of the Streams servlet `web.xml` file, as follows. Note that line breaks have been added to fit the example on the page.

```
<init-param>
 <param-name>PluginClassNames</param-name>
  <param-value>com.ibm.unica.detect.interact.servlet.
  StreamServletRealTimePluginImpl.com.company.ClassName</param-value>
 </init-param>
```

Code examples

Your plugin must implement the `StreamServletRealTimePlugin` interface, which has only one function: `processRealtimeResponse`.

Note: In the following examples, line breaks have been added to fit the examples on the page.

Here is an example of code that implements the interface.

```
package com.ibm.unica.detect.interact.servlet;
 import java.util.Map;
  @SuppressWarnings("rawtypes")
   public interface StreamServletRealTimePlugin {
   public void processRealTimeResponse(
    final String audienceId,
    final String audienceLevel,
    final String componentId,
    final long timestamp,
    final Map response);
   }
```

Here is an example of a plugin implementation that prints all output parameters into the servlet log file.

```
 package com.ibm.unica.detect.interact.servlet;
  import java.util.Map;
  import org.apache.log4j.Logger;
  @SuppressWarnings("rawtypes")
   public class StreamServletRealTimePluginImpl
    implements StreamServletRealTimePlugin {
   private static Logger logger =
    Logger.getLogger(StreamServletRealTimePluginImpl.class);
   public void processRealTimeResponse(
   final String audienceId,
   final String audienceLevel,
   final String componentId,
   final long timestamp,
   final Map response)
   {
   logger.info("audienceLevel: " + audienceLevel + ";
   audienceId: " + audienceId + ";
   componentId: " + componentId + ";
   timestamp: " + timestamp + ";
   processRealTimeResponse: " + response);
    }
   }
```

processRealtimeResponse

`processRealtimeResponse (audienceId, audienceLevel, componentId, timestamp, response)`

audienceId

The value in the audience ID field in your customer data source.

This field must be defined as an audience field in Opportunity Detect.

audienceLevel

The audience level code assigned in Opportunity Detect to the audience level used in the transaction that produced the Outcome.

componentId

The ID of the Action component that produced the Outcome.

timestamp

The timestamp when the Action component fired its Outcome.

response

A map of the Outcome data that was specified in the **Message** and **Additional Information** fields of the Action component.

Chapter 9. Monitoring Opportunity Detect deployments on Streams

Opportunity Detect provides a monitoring tool for deployments running on Streams.

The monitoring tool runs as a JConsole service.

The data in the monitoring tool is supplied by the Streams Remote Control Service. Only running deployments and services are shown.

You can view the following information in real time.

- Health of the Streams jobs
- Metrics for individual Opportunity Detect deployments within Streams
- Aggregated metrics for servers, server groups and Opportunity Detect deployments

Notifications

Two types of notification are available.

- A Streams service or job (deployment) failure can trigger a notification to users who have Opportunity Detect roles. Users who have Opportunity Detect roles and who subscribe to Marketing Platform system alerts receive these notifications, which include details, in their IBM Marketing Software inbox.

  If email notifications are configured, subscribing Opportunity Detect users also receive an email for each failure.
- Users of the monitoring tool can also subscribe to notifications that appear in the tool, using the JConsole notifications feature.

Problem diagnosis

The monitoring tool helps you to diagnose problems by providing information on the following types of failure.

- Streams Remote Control Service (RCS)

  The Streams RCS obtains metrics. There is a single instance of the RCS across all server groups. After an RCS failure, the RCS is unable to obtain metrics.
- Streams Real Time Web Connector (RTC)

  The RTC is the incoming gateway for real-time traffic. There is a single instance of the RTC for each server. After an RTC failure, the load balancer routes incoming traffic to other RTC instances that are running.
- Streams Real Time Proxy (RTP)

  There is a single instance of the Streams RTP for each server. The RTP routs incoming messages to engines. An RTP failure causes loss of incoming messages.
- Database failure

  When there is a database failure, engines fail to update State History. This causes engines to fail when they exceed their in-memory cache limit or the number of allowed connection retries.
- Engine failure

After an engine failure, transactions targeted to the failed engine are no longer processed.

- Streams input/output jobs

  Input/output jobs are used in the operation of the Opportunity Detect batch feeder and listener. After a Streams failure, parts of the input/output chain fail.

Setting database connection properties for monitoring

Two files contain the database connection information required by the monitoring tool.

About this task

- `jdbc.properties` contains database user credentials and JDBC configuration for the connection to the Marketing Platform database, for Marketing Platform notifications.
- `monitor.xml` contains database user credentials and JDBC configuration for the connection to the Opportunity Detect design time database

You must edit these files in the following cases.

- If Opportunity Detect is installed using the manual database creation option, you must set values in both of these files manually.

  If the automatic database creation option is used, and if Opportunity Detect Design Time and Run Time are installed on the same server, the installer sets the values.
- If Opportunity Detect Design Time and Run Time are installed on different servers, you must set values in the `monitor.xml` file.
- If Opportunity Detect is installed using the manual database creation option, and if Opportunity Detect Design Time and Run Time are installed on different servers, the files are laid down in DB2 format.

  However, sample Oracle files are provided in the `conf` directory under your Opportunity Detect Run Time installation. If you are using Oracle for your system tables in this scenario, you must do the following.

  1. Change the names of the `jdbc.properties` and `monitor.xml` files.

     You can use any name, such as `monitor.xml.old`.
  2. Rename the `jdbc.properties.oracle` file to `jdbc.properties`.
  3. Rename the `monitor.xml.oracle` file to `monitor.xml`.

  Then proceed with the steps shown in this procedure.

Procedure

1. In the **conf** directory under your Opportunity Detect Run Time installation, locate the `jdbc.properties` file, open it in a text editor, and do the following.
   a. Set the value of the `uasm.data_source_password` property to the password of the Marketing Platform database user account.

      For example: `uasm.data_source_password=YOUR_PASSWORD`

      You can encrypt this password by running the `encode` shell script located in the `cli` directory under your Opportunity Detect design time installation. For example: `./encode.sh YOUR_PASSWORD`

      If you use an encoded password, precede the encoded password with ENC and enclose the password in parentheses. For example: `uasm.data_source_password=ENC(15jqpwXzPPor4FChvS63Gw==)`
   b. Save and close the file.

2. In the **conf** directory under your Opportunity Detect Run Time installation, locate the `monitor.xml` file, open it in a text editor, and do the following.
   a. Set the value of the `ConnectionPassword` key to the password of the Opportunity Detect database user account.

      For example: `<entry key="ConnectionPassword">YOUR_PASSWORD</entry>`

      You can encrypt this password as described in the previous step.

      If you encrypt the password, set the `ConnectionPasswordEncrypted` key to `true`. For example:

      ```
      <entry key="ConnectionPassword">hJOvSd18EqBFPWr2+a36Ag==</entry>
      <entry key="ConnectionPasswordEncrypted">true</entry>
      ```

   b. Save and close the file.

Monitoring tool configuration

You use JConsole to connect to the monitoring service as a remote client. Optionally, you can set configuration properties to configure the polling interval and the length of time that monitoring data is retained, logging levels, and the port that the tool uses.

Poll interval and data retention

To control the poll interval and the length of time that monitoring data is retained, set the values of the following two properties on the **Settings > Configuration** page under **IBM Opportunity Detect and Interact Advanced Patterns | System | Monitoring**.

- `Poll Interval` - The number of seconds that the service waits between two successive polls of the Streams server for the statistics. The default is 300 seconds, or 5 minutes.
- `Retaining Time (in days)` - The number of days the service keeps the polled data in the database. The default is 10 days. The number of days the monitoring service should keep the polled data in the database. The default is 10 days. Data that is older than the time specified here is purged.

Logging

The `log4j.properties` file has logging properties for the `monitor.log` file. The`log4j.properties` is located in the `conf` directory under your Opportunity Detect Run Time installation.

The `monitor.log` file is located in the `logs` directory under your Opportunity Detect Run Time installation.

JMX configuration

JMX configuration values are passed in the monitoring tool startup script located in the **/home/streamsadmin/OpDetection/monitor** directory under your Opportunity Detect Run Time installation. Remote JMX clients use this information to connect to JMX interface in the monitoring tool. You can modify these values to start the JMX service on the desired network port.

Open the `monitor.sh` file in a text editor.

Modify the port number in the following lines.

```
JMX_ARGS = -Dcom.sun.management.jmxremote=true \
-Dcom.sun.management.jmxremote.port=9999 \
-Dcom.sun.management.jmxremote.authenticate=false \
-Dcom.sun.management.jmxremote.ssl=false
```

Related reference:

"IBM Opportunity Detect and Interact Advanced Patterns | System | Monitoring" on page 117

Starting the Opportunity Detect monitoring service

The Opportunity Detect monitoring tool runs as a JConsole service. After you start the service, you can leave it running.

Procedure

1. Log in to the Run Time machine as the streamsadmin user.
2. Start the monitoring service by running `monitor.sh`

   The `monitor.sh` script is located in the `OpDetection/monitor` directory under your Opportunity Detect Run Time installation.

Viewing Opportunity Detect monitoring data

The Opportunity Detect monitoring tool runs as a JConsole service. To view the monitoring data, you must have the Java Development Kit (JDK) installed.

Before you begin

The JConsole executable can be found in the `bin` directory under your JDK installation.

If this directory is in your system path, you can start JConsole by typing `jconsole` in a command (shell) prompt. Otherwise, you have to type the full path to the executable file in a command prompt.

For complete details on using JConsole, see http://docs.oracle.com/javase/6/docs/technotes/guides/management/jconsole.html.

Obtain the following information.

- The host and port configured for the monitoring service.
- Valid user credentials on the Run Time machine. If LDAP authentication has been enabled, you can use valid LDAP user credentials.

Procedure

1. Start JConsole.
2. Select **Remote Process** and complete the fields as follows.
   - Remote Process: Enter the configured *host:port* of the monitoring service.
   - Username: Enter the user name you obtained.
   - Password: Enter the password.
3. On the MBeans tab, in the left panel, under the **com.ibm.detect.monitor** node, expand any of the nodes and drill down until you see a node that has a Java bean icon  .

The nodes with Java bean icons are parents of nodes that show data about the selected server group, server, deployment, or Streams service.

Related reference:

"Metrics in the monitoring tool"

"Streams jobs" on page 82

Metrics in the monitoring tool

The Attributes fields in the monitoring tool provide details that can help you to determine the health of the Opportunity Detect runs being executed on your Streams servers.

The monitoring tool can show metrics at the server group, server, deployment, and component level. The following table describes the fields shown in the Attributes node.

The polling interval and retention period mentioned in the following table are configurable on the **Settings > Configuration** page under **IBM Opportunity Detect and Interact Advanced Patterns | System | Monitor**.

Table 24. Monitoring metrics

Field	Description
Wider Interval	An interval of time you can set to see the data over a longer period than the polling interval. Persisted data from the database is used for this calculation. The average TPS informs you of throughput at the last polling; but it gives you no perspective on whether that number is normal or slow. The wider interval gives you a broader perspective. Every time an average TPS is calculated it is written to the database, so the tool has access to historical average TPS values. When you specify a wider interval, the tool averages all the historical TPS values for that time period. In this way, you have two numbers to compare: the current TPS against the historical average. You should set the wider interval value to less than the retention period.
Average tps	The average number of transactions per second processed during a polling interval. Persisted data from the database is used for this calculation.

Table 24. Monitoring metrics (continued)

Field	Description
Health Status	Health is aggregated over all health statuses of all components. Failure of an engine causes the server to become unhealthy, which in turn causes a deployment to become unhealthy. A deployment involves other jobs that handle input and output and potentially several engines. When polling is done, the tool looks at all these jobs and determines whether they are healthy. If all are healthy, this value is healthy; if any one is unhealthy, this value is unhealthy. The possible values are: • HEALTHY • PARTIALLY HEALTHY, • PARIALLY UNHEALTHY • UNHEALTHY
Transactions per second (tps)	The number of transactions processed per second over the polling interval by a deployment, server, or engine.
Total transactions	The total number of transactions processed over the polling interval.
High Engine tps/Low engine tps	Transactions per second over the polling interval on the highest performing engine among all engines on the server. Like health status, high and low engine TPS are aggregate values across engine transaction processing rates belonging to that server or deployment. High engine TPS represents the highest TPS value for any engine recorded in the last poll. Low engine TPS represents the lowest value. These values give you insight into the overall behavior of the system. For example, the average TPS in the context of the wider interval TPS might be okay; but some engine may be slow performing. By contrasting the high and low TPS, you the gain insight that there is a slow performing engine.

Related tasks:

"Viewing Opportunity Detect monitoring data" on page 80

Streams jobs

The monitoring tool displays the health of the Streams jobs running for the various deployments.

The following table describes the way the various Streams jobs function in Opportunity Detect.

Table 25. Streams jobs

Job	Description
LegacyOutcome	Sends Outcomes to tables.
Engine	Executes workspace logic.

Table 25. Streams jobs (continued)

Job	Description
Feeder	Reads batch feed files and sends them to the engine.
RealTimeProxy	A gateway or router job that sends incoming messages to RealTimeInputOutput
RealTimeInputOutput	Reads real time input messages from RealTimeProxy and sends them to the engine. If there are Outcome messages it sends them to the web service.
RealTimeFeeder	Continuously watches the directory designated for real time files, reads them and sends them to engines.
RunController	Controls batch runs by sending commands to UnicaFeeder and engines.
TimeQueue	A manager for sending EETQ signals for deployments that include negative Pattern or Forward Inactivity components.
InteractConnector	Reads messages from the Interact service, sends the messages to the engine, and sends the resulting Outcomes.to the Interact service.

Related tasks:

"Viewing Opportunity Detect monitoring data" on page 80

Connectors and Streams jobs

Depending on the data source connectors used in a deployment, different Streams jobs and services perform the required processing.

The following table maps the various combinations of input and outcome connectors to the Streams jobs that you can expect to see in the monitoring tool when a deployment runs.Opportunity Detect.

Note that the CommAgent and TimeQueue (Unica EETQ) jobs are not related to file connectors. These jobs may appear with any of the connector combinations shown in the table, depending on the trigger system.

Table 26. Connectors and Streams jobs

Input connector	Outcome connector	Jobs and services
Batch file	Table	• RunController • Feeder • LegacyOutcome • Engine
Real time file	Table	• RealTimeProxy • UncaRealTimeFeeder • LegacyOutcome • Engine
Queue	Queue	• Engine
Queue	Table	• LegacyOutcome • Engine

Table 26. Connectors and Streams jobs (continued)

Input connector	Outcome connector	Jobs and services
Web Service	Web Service	• RealTimeProxy • RealTimeInputOutput • Engine
Web Service	Table	• RealTimeProxy • RealTimeInputOutput • LegacyOutcome • Engine
InteractConnector	InteractConnector	• Engine • InteractConnector • RealTimeProxy

Setting the wider interval

Follow these steps to set the wider interval for any deployment shown in the Opportunity Detect monitoring service.

Procedure

1. Under the **Deployment** node, expand a server group.

   All the deployments in the server group are listed.
2. Expand a deployment, expand **Operations** and click **setWiderIntervalInSeconds**.
3. In the **param1** field, enter the number of seconds you want for the wider interval.

   You should set the wider interval value to less than the retention period set in the **Retaining Time (in days)** configuration property on the **Settings > Configuration** page under **IBM Opportunity Detect and Interact Advanced Patterns | System | Monitoring**.
4. Click **setWiderIntervalInSeconds**.

Results

The wider interval you set is applied. You can see wider interval metrics for any deployment under the **Attributes** node.

Configuring monitoring notifications

Follow these steps to configure notifications for the Opportunity Detect monitoring service.

About this task

There are two types of monitoring notifications.

- Marketing Platform can deliver notifications to the IBM Marketing Software inbox of users who have Opportunity Detect roles and who have subscribed to Marketing Platform system alerts.
- Notifications are available within the monitoring tool under the various nodes.

Procedure

1. To have Marketing Platform send notifications to Opportunity Detect users, do the following.
   a. Under the **Deployment** node, expand a server group.

      All the deployments in the server group are listed.

   b. Expand a deployment, expand **Operations** and click **enableMarketingPlatformNotifications**.

      You can check the Marketing Platform notification setting for any deployment under the **Attributes** node. The **MarketingPlatformNotificationsEnabled** attribute is set to **true** when notifications are enabled for a deployment.

      Users who subscribe to Marketing Platform system alerts receive the notifications. Users can subscribe for themselves, or an administrator can subscribe for them.
2. To receive notifications within the monitoring tool, under any **Deployment**, **Service**, or **StreamsComponent** node, expand **Notifications** and click **Subscribe**.

   After you subscribe, notifications appear in the **Notifications** node.

Related tasks:

"Configuring email notifications in IBM Marketing Software"

Configuring email notifications in IBM Marketing Software

Follow this procedure to configure the Marketing Platform to send system alert and notification emails to users. You must have an email server set up before you start.

Before you begin

Obtain the following information about your mail server.

- The protocol used by your mail server.
- The port on which the mail server listens.
- The name of the machine that hosts your mail server.
- Whether your mail server requires authentication.
- If your mail server requires authentication, an account name and password on the mail server.

About this task

Tip: See the related references if you need additional details about performing this procedure.

Procedure

1. If your mail server requires authentication, save a mail server account name and password as a data source in a Marketing Platform user account.

   Use an internal Marketing Platform user account, not a user imported from an LDAP server.

   Make a note of the Marketing Platform user name and the data source name, as you will use them in step 3.

2. Log in to IBM Marketing Software as a user with administrative privileges in Marketing Platform.
3. On the **Settings > Configuration** page, set the configuration properties in the following categories.
   - `General | Communication | Email`
   - `Platform | Notifications`

   Use the information you obtained about your mail server to set values.

Related tasks:

"Configuring monitoring notifications" on page 84

Related reference:

"IBM Marketing Software | General | Communication | Email" on page 119

"IBM Marketing Platform | Notifications" on page 120

Chapter 10. Automating tasks using the RemoteControlCLI utility

The Opportunity Detect command line utility, `RemoteControlCLI` (CLI), allows you to automate the management of deployment configurations and batch runs. You can use the CLI to perform the same actions you can perform on the Deployment & Batch Run tab of a workspace.

The `RemoteControlCLI` utility and sample batch or shell scripts are all located in the `cli` directory under your Opportunity Detect design time installation.

The following sample scripts are provided with your installation.

Table 27. Command line scripts

Script	Usage
`encode`	Encrypt the password that the CLI uses to connect to the design time database.
`deploy`	Deploy a deployment configuration.
`stop`	Stop the deployment of a deployment configuration.
`start`	Re-start the deployment of a deployment configuration.
`startBatch`	Start a batch run by starting a deployed deployment configuration.
`stopBatch`	Stop a batch run by stopping a deployed deployment configuration.

Prerequisites

The following are prerequisites for using the Opportunity Detect command line scripts.

- The computer where you run the scripts must have network access to machines where the Opportunity Detect design time and run time components are installed.
- Java version 7 must be installed on the machine where you run the scripts.
- You must set the `JAVA_HOME` system environment variable. You can do this either on the machine where you run the scripts, or by editing the scripts to set it temporarily when the scripts run.

Password security

The information the CLI uses to connect with the design time database is saved in the `RemoteControlCLI.xml` file, located in the `cli` directory. This information includes the user name and password for an account in the database.

You can encrypt the password that is stored in this file, to avoid storing the password in clear text.

Setting up the database connection for the Opportunity Detect CLI

You edit the `RemoteControlCLI.xml` file to enable the CLI to connect with the Opportunity Detect design time database.

Before you begin

Obtain the following information about your design time database.

- The name and URL of the database used with your design time installation.
- The user name and password for an account with Administrator privileges in the design time database.
- The URL of your design time installation
- The URL and class name for the database driver used with your design time database.

Procedure

1. Open the `RemoteControlCLI.xml` file, located in the `cli` directory under your Opportunity Detect design time installation.
2. Run the `encode` script to encrypt the password for the database account. Copy the resulting string and use it as the value for the `ConnectionPassword` key in the `RemoteControlCLI.xml` file.

   For example: `./encode.sh YOUR_PASSWORD`

   This step is optional, but provides the best security. If you do not encrypt the password, it is stored in clear text in the `RemoteControlCLI.xml` file.

   If you use an encrypted password, set the value of the `ConnectionPasswordEncrypted` key to `True`.
3. Use the database information you obtained to complete the values in the rest of the keys in the `RemoteControlCLI.xml` file.
4. Save and close the `RemoteControlCLI.xml` file.

Results

You can now run the CLI commands.

Related reference:

"RemoteControlCLI XML reference"

RemoteControlCLI XML reference

The values you enter in the `RemoteControlCLI.xml` file enable the CLI to connect with your Opportunity Detect design time database.

Table 28. Reference for CLI XML keys

XML key	Value
RemoteControlURL	The URL for your Opportunity Detect design time installation. Example: `http://example.com:8080/axis2/services/RemoteControl`
ConnectionDriverName	The class name of the database driver used with your Opportunity Detect design time installation. Example: `com.ibm.db2.jcc.DB2Driver`
ConnectionURL	The URL of the driver used with your Opportunity Detect design time installation. Example: `jdbc:db2://example.com:50000/Detect91:retrieveMessagesFromServerOnGetMessage=true;`

Table 28. Reference for CLI XML keys (continued)

XML key	Value
ConnectionUserName	The user name of an account in the Opportunity Detect design time database. Example: `example_user_name`
ConnectionPassword	The password of the same account used for the ConnectionUserName. Example: `example_password`
ConnectionPasswordEncrypted	A flag that indicates whether the password value used for ConnectionPassword has been encrypted using the `encode script`. If you use an encoded password, set this value to `True`.
Schema	The URL of the Opportunity Detect design time database. Example: `http://example.com:8080/axis2/services/RemoteControl`

Related tasks:

"Setting up the database connection for the Opportunity Detect CLI" on page 87

RemoteControlCLI command reference

The RemoteControlCLI utility supports the following commands and options.

Syntax

```
RemoteControlCLI deploy -d deployment configuration ID -v version number

RemoteControlCLI start -d deployment configuration ID -v version number

RemoteControlCLI stop -d deployment configuration ID -v version number

RemoteControlCLI startBatch -d deployment configuration ID -v deployment
configuration version number -w workspace ID -fp path to feed files
[-am Off|EndOfDay|EndOfRun ]
[-ll Off|Fatal|Error|Warn|Info|Debug|Trace]
[-n notification file name]
[-r] [-ri] ID of the batch run to recover
[-im Off|On]
[-id inactivity date in YYYY-MM-DD HH:MM:SS format]

RemoteControlCLI stopBatch -d deployment configuration ID -v version number
```

You can obtain the help for this utility by entering `-h` by itself or with any of the above commands.

Where to find the deployment configuration ID and version number

You can obtain the version number and ID of the deployment configuration on the Deployment tab of the workspace. Select the deployment configuration and select the History tab in the panel that opens. Look under the Version and Message columns for the version and configuration ID. When you run the command, increment the deployment configuration version by 1.

For example, if the deployment version is currently `n`, you would specify a deployment configuration version number of `n+1`.

Command and option details

RemoteControlCLI deploy -d *deployment configuration ID* -v *deployment configuration version number*

Deploy a deployment configuration to the Streams server. For a first time deployment, you must use the Deployment & Batch Run tab of the workspace. You can perform subsequent deployments using the CLI.

RemoteControlCLI start -d *deployment configuration ID* -v *deployment configuration version number*

Start a deployment configuration.

RemoteControlCLI stop -d *deployment configuration ID* -v *version number*

Stop a deployment configuration.

RemoteControlCLI startBatch -d *deployment configuration ID* -v *deployment configuration version number* -w *workspace ID* -fp *path to feed files*

Start a batch run.

The non-required options for this command allow you to set the same parameters that are available on the Deployment & Batch Run tab, as follows.

- `-am` *Off*|*EndOfDay*|*EndOfRun*
  Set the artificial transaction mode
- `-ll` *Off*|*Fatal*|*Error*|*Warn*|*Info*|*Debug*|*Trace*
  Set the logging level for all Streams components
- `-n` *notification file name*
  Set the file used to send notifications about run success or failure
- `-r`
  Run in recovery mode (requires `-ri`)
- `-ri` *ID of batch run*
  Set the ID of the batch run to recover (used with `-r`)
- `-im` *Off*|*On*
  Turn inactivity mode on or off. If you do not set this mode, the default is `off`.
- `-id` *inactivity date in YYYY-MM-DD HH:MM:SS format*
  Set the date for the inactivity mode (used with `-im`). 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.

RemoteControlCLI stopBatch -d *deployment configuration ID* -v *version number*

Stop a batch run.

Chapter 11. Exporting and importing workspace logic with the export/import utility

Use the export/import utility to transfer Opportunity Detect workspace logic from one environment to another.

The export/import utility operates at the workspace level. You can export all workspaces or specify the workspaces you want to export.

The utility creates an XML file that contains a specification of the workspace logic and all of its dependencies. The export includes all the defined data sources, audience level codes, and named value lists regardless of their usage in the exported workspace.

After export, you can transfer the XML file to the target environment and import the logic. After an export/import cycle, the logic transferred is functionally equivalent in both environments.

The active to passive model

The utility supports only an active-to-passive environment model. In this model, the following assumptions are made.

- The source environment that generates the export is the only environment in which active development occurs.
- No regular development occurs in the target environment where the logic is imported.
- If development occurs in the target, the changes must be applied manually to the source environment so that the changes are not lost in the next export/import cycle

If changes are not made in the target environment between export/import cycles, the import process creates new a workspace in the target if the imported workspace does not exist, or overwrites the workspace in the target if the imported workspace does exist.

If changes are made in the target environment between export/import cycles, conflict resolution rules are applied.

Password security

The information the export/import utility uses to connect with the design time database is saved in an XML file located in the `tools/librarymgr/conf` directory under your Opportunity Detect installation. This information includes the user name and password for an account in the database. As described in the setup procedures, you can encrypt the password that is stored in this file to avoid storing the password in clear text.

Logging

When the utility has finished running, you can check the log to verify the dependencies. For example, it is possible to import a workspace that includes components referenced in other workspaces. These dependencies are listed in the log file.

Draft comment
Not sure I got this right. What should we say about logging? Where is the log located? How is it configured?

Related concepts:

"Conflict resolution"

"Best practices for export / import" on page 93

Chapter 12, "Conversion utility (with FixPack 10.0.0.1 only)," on page 99

Conflict resolution

If changes were made to the target environment between two export/import cycles, then the following conflict resolution rules are applied during the import.

- Audience levels
  - If the target contains the same audience level name with a different ID, the name on the target is appended with "_Target".
  - If the target contains an audience level with a different name but same ID, the ID on the target is changed to a new unused code.
- Named value lists
  - If the target contains the same named value list name with a different ID, the name on the target is appended with "_Target".
  - If the target contains a named value list with a different name but same ID, the named value list on the target is overwritten with the source.
- Named value items, for a given named value list
  - If the target contains the same named value item name with a different ID, the name on the target is appended with "_Target".
  - If the target contains a named value item that does not exist on the source and is not used by components on the target, the item is removed.
- Data sources
  - If the target contains the same data source name with a different ID, the name on the target is appended with "_Target".
  - If the target contains a data source with a different name but same ID, the data source on the target is overwritten with the source.
- Data source fields, for a given data source
  - If the target contains the same data source field name with a different ID, the name on the target is appended with "_Target".
  - If the target contains a data source field that does not exist on the source and is not used by components on the target, the field is removed.
- Workspaces
  - If the target contains the same workspace name with a different ID, the name on the target is appended with "_Target".

  - If the target contains a workspace with a different name but same ID, the Workspace on the target is overwritten with the source.
- Components, for a given workspace
  - If the target contains a component that does not exist on the source and is not used by components on the target workspace, the component is removed from the target workspace.
  - If the target contains a component with the same ID, the component on the target is overwritten with the source.
  - Consider the following scenario.
    - A source environment contains a workspace with a component that is referenced by another workspace.
    - An export/import cycle brings the workspace containing the origin component into the target environment, but does not import the workspace where a reference to that component was used.
    - In a subsequent cycle, the workspace containing the component reference is imported.

    In this scenario, a component reference is created in the target environment.

Best practices for export / import

Follow these guidelines when exporting and importing workspace logic.

Exporting and importing collector workspaces

A workspace can include multiple trigger systems, but it is a good practice during the development and test phase to use a different workspace for each trigger system.

When you have tested all of your trigger systems in the development environment, you can then use component references to copy them to one collector workspace.

You can then export the collector workspace, import it into your production environment, and deploy it. Because the utility includes dependencies in the export, there is no need to import all of the workspaces referenced by the collector workspace.

Partial imports

You might import only a subset of your development workspaces. If a workspace that was affected by changes made in development exists in your production environment but is not updated during an import, you should not redeploy it.

The old deployment of the affected workspace should continue to function as usual, unless data sources used by that workspace have changed.

Changes in the production environment

In some exceptional circumstances, it might be necessary to make a change to a component in the production environment. If you do this, be sure to make the same change in the development environment so that your change is not overwritten during the next export/import cycle.

If you make a change in a stateful component such as a Pattern in your production environment, and you make same change in development, note that the state of

the component will be different in the two environments.

Setting up to export workspace logic

Use this procedure to set up to export Opportunity Detect workspace logic from your source environment.

Before you begin

The export/import utility requires IBM Java 1.7.

If the source or target Opportunity Detect design database is Oracle, ensure that `ojdbc7.jar` is added to your classpath before you run the export/import utility.

Procedure

1. Edit the `librarymanager` file in your source environment to set the *JAVA_HOME* variable to reflect the location of IBM Java 1.7 in your source environment.

   The utility is a script. It has the `.bat` extension on Windows and the `.sh` extension on UNIX type systems.

   This file is located under your Opportunity Detect installation in the `tools/librarymgr` folder.
2. In the `tools/librarymgr/conf` directory under your Opportunity Detect installation, open the appropriate XML file for your database type, as follows.
   - If your database type is DB2, open the `importexport.xml` file.
   - If your database type is Oracle, open the `importexport.xml.oracle` file.
3. Edit the file you opened in the previous step to reflect the database used in your target environment.

   Run the encode script located in the `cli` directory under your Opportunity Detect installation to encrypt the password for the database account. Copy the resulting string and use it as the value for the `ConnectionPassword` key in the XML file.

   This step is optional, but provides the best security. If you do not encrypt the password, it is stored in clear text in the XML file. If you use an encrypted password, set the value of the `ConnectionPasswordEncrypted` key to `True`.

   Draft comment
   Has an encrypted password been tested with this utility? I wonder how the password gets decrypted?
4. Save the file.

   If your database type is Oracle and you edited the `importexport.xml.oracle`, rename the file as `importexport.xml`.

What to do next

You can now run the utility to export workspace logic.

Related reference:

"Export/import utility command reference" on page 96

"Export/import XML reference" on page 95

Setting up to import workspace logic

Use this procedure to set up to import Opportunity Detect workspace logic to your target environment.

Before you begin

- The export/import utility requires IBM Java 1.7.
- Before you can import workspace logic, you must run the export/import utility in your source environment to produce the XML file containing the workspace logic.
- If the source or target Opportunity Detect design database is Oracle, ensure that `ojdbc7.jar` is added to your classpath before you run the export/import utility.
- It is a good practice to back up your Opportunity Detect system tables before performing an import.

Procedure

1. Edit the `librarymanager` file in your source environment to set the *JAVA_HOME* variable to reflect the location of IBM Java 1.7 in your target environment.

   The utility is a script. It has the `.bat` extension on Windows and the `.sh` extension on UNIX type systems.

   This file is located under your Opportunity Detect installation in the `tools/librarymgr` folder.
2. Copy the XML file containing the exported workspace logic from your source environment and place it in the directory in your target environment where the `librarymanager` script is located.

What to do next

You can now run the utility to import workspace logic.

After the import is complete, deploy or redeploy your imported workspaces.

Related reference:

"Export/import utility command reference" on page 96

"Export/import XML reference"

Export/import XML reference

The values you enter in the export/import XML file enable the export/import utility to connect with your Opportunity Detect design time database.

Table 29. Reference for export/import XML keys

XML key	Value
RemoteControlURL	The URL for your Opportunity Detect remote control service. Example: `http://RCS_SERVER:RCS_PORT/axis2/services/RemoteControl`

Table 29. Reference for export/import XML keys (continued)

XML key	Value

ConnectionDriverName	The class name of the database driver used with your Opportunity Detect design time installation.
	
	Examples:
	• DB2:
	`com.ibm.db2.jcc.DB2Driver`
	• Oracle:
	`oracle.jdbc.OracleDriver`

ConnectionURL	The URL of the driver used with your Opportunity Detect design time installation.
	
	Examples:
	• DB2:
	`jdbc:db2://HOST:PORT/Detect91:retrieveMessagesFromServerOnGetMessage=true;`
	• Oracle:
	`jdbc:oracle:thin:@//HOST:PORT/DATABASE_NAME`

ConnectionUserName	The user name of an account in the Opportunity Detect design time database.
ConnectionPassword	The password of the account used for the ConnectionUserName.
ConnectionPasswordEncrypted	A flag that indicates whether the password value used for ConnectionPassword has been encrypted. If you use an encoded password, set this value to `True`.
Schema	The name of the Opportunity Detect design time schema.

Draft comment
This is based on the Oracle version of this file. Does the non-Oracle version contain any different XML tags?

Related tasks:
"Setting up to export workspace logic" on page 94
"Setting up to import workspace logic" on page 95

Export/import utility command reference

The export/import utility supports the following commands and options.

Syntax

The utility is a script. It has the `.bat` extension on Windows and the `.sh` extension on UNIX type systems. Examples are shown for Windows.

`librarymanager.bat export -f *exportFileName.xml*`

Export all workspaces in the source environment. The exported XML file is placed in the directory where the `librarymanager.bat`file is located.

Use the **-w** option to export only specified workspaces in the source environment, as follows.

```
librarymanager.bat export -f exportFileName.xml -w
workspace_name1,"workspace name 2"
```

If there are spaces in the workspace name, enclose the workspace name in quotes. For example, `"workspace name n"`

There should be no spaces before or after the comma separating two workspace names.

librarymanager.bat import -f *exportFileName.xml*

Import workspaces to the target environment.

Related tasks:

"Setting up to export workspace logic" on page 94

"Setting up to import workspace logic" on page 95

Chapter 12. Conversion utility (with FixPack 10.0.0.1 only)

The conversion utility is available only if you have applied Opportunity Detect FixPack 10.0.0.1. The conversion utility enables you to convert workspaces in Unica Detect format (that is, in versions earlier than 9.1.1) to Opportunity Detect format.

The conversion utility works with workspace logic only; data migration from pre 9.1.1 versions is not supported.

The conversion utility works in conjunction with the export/import utility, as follows.

1. Use the export/import utility to export workspace logic from your old version of Unica Detect.

   The export/import utility creates an XML file that contains a specification of the workspace logic and all of its dependencies. The export includes all the defined data sources, audience level codes, and named value lists regardless of their usage in the exported workspace.
2. Apply the conversion utility to the XML file generated in the first step.

   The conversion utility generates an XML file compatible with the Opportunity Detect format.
3. Use the export/import utility to import the XML file generated by the conversion utility into Opportunity Detect.
4. Test your import as follows.
   - Consult the conversion log and follow up with manual changes when an issue is encountered.

     Draft comment

     Where is this log located? What is the log name? What are the issues and what action does the user take for each one?
   - Open each imported workspace and check to make sure all components are present and that the imported workspace matches the original one.
   - Perform any required manual cleanup.
   - Perform a test run of the imported workspaces.

Post-import manual cleanup

Because of differences between pre-9.1.1 versions of Unica Detect and Opportunity Detect, some manual steps might be required after you import the converted XML. The following table describes the items that require manual steps.

Table 30. Manual cleanup items

Item	Manual step
Presentation layer lists in pre-9.1.1 versions are called Named value lists Opportunity Detect. Pre-9.1.1 versions allowed date values, which are not supported in Opportunity Detect.	Use a Date expression in your workspace to work around this limitation.
Time constants in pre-9.1.1 versions include Beginning of Next Month, Beginning of N Months Ago, which do not exist in Opportunity Detect.	Replace these time constants with Date expressions. For example, Beginning of next month = Date Expression (Beginning of this month + 1 month).
Select components in pre-9.1.1 versions supported the Join function, which is not supported in Opportunity Detect.	**Draft comment** What is the workaroound for this?
Operators NAND and NOR are not available in 10.0.	Equivalencies of NAND, NOR can be built with Expression Builder.
Nested comparison is not available in 10.0.	Mixed expressions have to be built in Boolean expressions.
Subgroups in Trend components are not available in 10.0.	You can achieve a similar result using Trends components in conjunction with aggregation in Container components.
Labels in the Additional Outcome fields of the Action component are optional in version 8.5, but they are mandatory in Opportunity Detect.	The conversion utility adds labels with default names (label_n). If you want more descriptive labels, you must edit these label names.
The file names of 8.5 workspaces are different than 10.0 workspaces.	Manually change file name using these guidelines. • File names in Opportunity Detect are case sensitive. • Consult the *Opportunity Detect Administrator's Guide* to update file names in compliance with the new naming convention.
Roles and permissions are not the same in versions 8.5 and 10.0.	The following roles are applied to converted users. • The View permission in 8.5 becomes OpDetectViewer role in 10.0. • The Modify permission in 8.5 becomes OpDetectAdmin role in 10.0. • The Run permission in 8.5 becomes OpDetectProductionDesigner role in 10.0. In 10.0, permissions are not implemented at the object level (for example, workspace ownership).

Related concepts:

Chapter 11, "Exporting and importing workspace logic with the export/import utility," on page 91

Chapter 13. Troubleshooting Opportunity Detect

Some suggestions for resolving issues with Opportunity Detect are listed in this section, along with workarounds or solutions.

Obtaining Streams logs

When a problem occurs with a Opportunity Detect deployment, you should examine the Streams log to help diagnose the problem.

About this task

Note: To include more detail in the Streams log, set the system log level to Debug on the Batch Run tab of a workspace deployment.

Procedure

1. In Opportunity Detect, go to **Detect > Workspace Manager** and select the workspace that is having the problem.
2. Obtain the ID of the deployment configuration. On the Deployment tab of the workspace, select the deployment configuration and select the History tab in the panel that opens. The configuration ID is listed in the Message column.
3. On the Streams sever, navigate to the following directory.

   `/home/streamsadmin/OpDetection/deploy/`*`Deployment_Configuration_ID`*`/current`
4. Open the Streams `job_ids.list` file and note the Streams job IDs.
5. Log in to Streams as the streamsadmin user and select the Streams job IDs you noted in the previous step. Click the **Download Trace File** icon.

   All the relevant logs are downloaded in compressed format.

How to verify that Opportunity Detect services are running

Symptoms

When deployment runs fail, a good first step is to verify that all the required services are running.

Resolving the problem

To verify that the Streams Remote Control, Interact Design, and Real Time Connector services are running, point your browser to the following URLs.

1. **Streams Remote Control**: `http://`*`HOST_NAME`*`:8080/axis2/services/RemoteControl?wsdl`, where *`HOST_NAME`* is the runtime server name.

   An XML page should be displayed.
2. **Interact Design**: `http://`*`HOST_NAME`*`:8181/axis2/services/InteractDesignService?wsdl`, where *`HOST_NAME`* is the design time server name.

   An XML page should be displayed.
3. **Real Time Connector**: `http://`*`HOST_NAME`*`:8282/servlets/StreamServlet?command=ping`, where *`HOST_NAME`* is the runtime server name.

   An OK message should be displayed.

Connection refused during Streams synchronization

Symptoms

When you synchronize Streams by clicking **Synchronize with changed Streams configuration** on the **Settings & Detect Settings > Server Group** page, you see this warning: "Connection refused."

Causes

The most likely cause is that the Streams RCS service is down.

Resolving the problem

Use the following command to start the StreamsRCS.

1. Log in to the Streams runtime server as the streamsadmin user.
2. Go to `/home/streamsadmin/OpDetection/StreamsRemoteControlService/wlp/bin`.
3. Execute the following command.

   ```
   ./server start StreamsRCS --clean
   ```

404 transport error during Streams synchronization

Symptoms

Problem: When you synchronize Streams by clicking **Synchronize with changed Streams configuration** on the **Settings & Detect Settings > Server Group** page, you see a popup with the error: "Transport error: 404 Error: Not Found."

Resolving the problem

In WebSphere, enable **Override class reloading settings** and set the **Polling interval for updated files** to 4 seconds. Then restart the WebSphere profile.

Streams dependency checker fails with ulimit warning

Symptoms

The Streams 3.2.1 dependency checker fails with a ulimit warning.

Resolving the problem

Increase the ulimit property as follows.

1. Log in as root to the Linux server on which you are trying to install Streams and edit the `90-nproc.conf` file located in the `/etc/security/limits.d` folder.

   Edit the file to have following entries

   ```
   # Default limit for number of user's processes to prevent
   # accidental fork bombs.
   # See rhbz #432903 for reasoning.

   *          soft    nproc    4096
   root       soft    nproc     unlimited
   ```
2. Restart the Linux server on which you are trying to install Streams.
3. Log in to Streams as the streamsadmin user.
4. Execute the following command.

   ```
   ulimit -a
   ```

   The outcome should include a line like this. `max user processes (-u) 4096`

MIN_TIMESTAMP feeder error in Streams log

Symptoms

After a workspace is deployed, the deployment configuration is down and some of streams jobs are unhealthy.

The following error appears in the Streams log: "com.ibm.unica.feeder/UnicaFeeder.spl:613:73: CDISP0053E ER-ROR: An unknown identifier was referenced in the SPL program: MIN_TIMESTAMP."

Causes
A date is present in profile data.

Resolving the problem
Dates in Profile data are not supported. Remove dates from your profile data.

Invalid XML error or CDISC5101E error in Streams log

Symptoms
One of the following errors appears in the Streams log.

- An invalid XML (Unicode : 0x45) was found in the prolog of the document
- The system cannot connect to the AAS service. See the previous error messages.

Resolving the problem
If you have version 9.1.1.2 or higher, set the JAVA_HOME environment variable as follows.

- Set the Streams instance and the StreamRCS service to use the Infosphere jre 1.7 on startup.
- Set the Realtime connector service to use the Infosphere jre 1.6 on startup.

  The StreamsRCS and Realtime Connector services are deployed on WebSphere Liberty profile application server. Typically the jre is set using JAVA_HOME or JRE_HOME environment variables in the `server.env` file in the liberty profile configuration directory. These environment variables are overridden if already set for the liberty profile instance.

  See the WebSphere documentation for more information on how to change the jre that the service uses: http://www-01.ibm.com/support/docview.wss?uid=swg21596484 for more information.

Deployment fails with STREAMS_ADAPTERS_ODBC environment variable error

Symptoms
Deployment fails with the following error in the trace log: "No STREAMS_ADAPTERS_ODBC environment variable set." This issue can occur when you use Oracle or DB2 for Profile tables, or Oracle for system tables.

Resolving the problem
Set the environment variables as described in the *IBM Opportunity Detect and Interact Advanced Patterns Installation Guide*, in the topic titled "Configuring the Streams database adapter for Oracle"

Workspace fails to deploy with boost library errors

Symptoms
The workspace fails to deploy. The trace log contains the following errors.

The trace log is located under your Streams installation in the /OpDetection/logs directory.

```
/home/streamsadmin/OpDetection/toolkits/unica_utils/impl/include/UnicaUtils.h:
        39:36: error: boost/iostreams/copy.hpp: No such file or directory
/home/streamsadmin/OpDetection/toolkits/unica_utils/impl/include/UnicaUtils.h:
        40:43: error: boost/iostreams/filter/zlib.hpp: No such file or directory
/home/streamsadmin/OpDetection/toolkits/unica_utils/impl/include/UnicaUtils.h:
        41:48: error: boost/iostreams/filtering_stream.hpp:
        No such file or directory
/home/streamsadmin/OpDetection/toolkits/unica_utils/impl/include/UnicaUtils.h:
        42:51: error: boost/iostreams/filtering_streambuf.hpp:
        No such file or directory
/home/streamsadmin/OpDetection/toolkits/unica_utils/impl/include/UnicaUtils.h:
        43:38: error: boost/iostreams/stream.hpp: No such file or directory
make: *** [build/function/com/ibm/unica/engine/GetFiringTime.o] Error 1
make: *** Waiting for unfinished jobs....
```

Resolving the problem

Install Boost libraries as described in the *IBM Opportunity Detect and Interact Advanced Patterns Installation Guide*, in the topic titled "Installing Boost libraries."

After installation, verify that the libraries are present under the /usr/include/boost directory.

JMS authenication fails

Symptoms

Runs of a workspace using a Queue connector fail with this error in the Streams log: "JMS Authentication Failed..."

Deployment of workspace using a Queue connector is successful, but the deployment configuration is down and some of Streams jobs are unhealthy.

Resolving the problem

Set variables in the streamsadmin .bashrc file, as described in the *IBM Opportunity Detect Administrator's Guide*, in the topic titled "Installing MQ client libraries."

Workspaces using a Web Service or Queue connector has unhealthy run

Symptoms

Workspaces using a Web Service or Queue connector deploy successfully but jobs (runs) are unhealthy. The following error is in the Streams log.

ERROR :::PEC.StartPE M[PECServer.cpp:runPE:207] P[5] - Could not load dll for PE 5 from /home/streamsadmin/OpDetection/deploy/ 2218dfd2-5a59-417b-bd5c-a17e07f6fd78/6/output/ com.ibm.unica.engine.UnicaEngine/Distributed/bin/com.ibm.unica.engine.UnicaEngine-a.dpe because of /data/databaseserver/DB210.1/lib64/libdb2.so.1: undefined symbol: sqlnlsFreeCpCvResources

Resolving the problem

1. Execute printenv LD_LIBRARY_PATH command as root and verify that the correct LD_LIBRARY_PATH value is set.

   If necessary, correct the value as described in the *IBM Opportunity Detect and Interact Advanced Patterns Installation Guide*, in the topic titled "Configuring the Streams database adapter for DB2" or "Configuring the Streams database adapter for Oracle."

2. Check the `.bashrc` file for both root and the Streamsadmin user to verify that the *Db2Instance* variable points to your database.
3. Restart the Streams instance and the Streams RCS Service.

With Oracle system tables, StreamsRCS fails to start

Symptoms

When you use Oracle for your system tables, the StreamsRCS service fails to start. In the StreamsRCS log the following error occurs.

```
 Error: Caused by: java.lang.RuntimeException: There were errors
initializing your configuration: <openjpa-2.2.1.1-SNAPSHOT-r422266:1431746
fatal user error> org.apache.openjpa.util.UserException: A connection could
not be obtained for driver class "oracle.jdbc.OracleDriver" and URL
"jdbc:oracle:thin:@//Machine-Name:Port/DB-Name". You may have specified an
invalid URL.
```

Resolving the problem

1. Copy the `ojdbc6.jar` file from your from Oracle installation to `/home/streamsadmin/OpDetection/StreamsRemoteControlService/wlp/usr/servers/StreamsRCS/dropins/StreamsRCS/WEB-INF/lib`
2. Restart the StreamsRCS and StreamsInstance services.

StreamsRCS service fails to start

Symptoms

The StreamsRCS service fails to start. The logs show the following details.

```
Launching StreamsRCS (wlp-1.0.2.cl0220130316-0213/websphere-kernel_1.0.2)
on IBM J9 VM, version pxa6470_27sr1fp1-20140708_01 (SR1 FP1) (en_US)
[AUDIT ] CWWKE0001I: The server StreamsRCS has been launched.
[err] Could not find matching export for Import-Package:
com.ibm.wsspi.logging; version="[1.0.0,1.1.0)"
[ERROR ] CWWKZ0005E: The application WEB-INF cannot start because
the server is not configured to handle applications of type StreamsRCS.
2015-11-24 00:56:07,645 [Default Executor-thread-2] INFO WarBasedAxisConfigurator
- Could not find axis2.xml,
loading default org/apache/axis2/deployment/axis2_default.xml from classpath
2015-11-24 00:56:07,697 [Default Executor-thread-2] INFO AxisServlet
- java.lang.IllegalArgumentException: The XMLInputFactory does not recognize
the property "reuse-instance".
[ERROR ] Uncaught.init.exception.thrown.by.servlet
```

Resolving the problem

Ensure that Java 6 is in your PATH environment variable.

Workspaces using a queue connector fail to deploy

Symptoms

A workspace that uses a Queue connector fails to deploy. The trace log contains an error such as the following.

```
Instantiate class: org.apache.activemq.jndi.ActiveMQInitialContextFactory
19 Nov 2015 18:16:39.758 [22294] ERROR #splapptrc,J[108],P[118],
ImportedTransactionSet_AKRT.TransactionSet_JMS_AKRT,spl_javaop M[JavaOp.cpp:
trace:82] - javax.naming.NoInitialContextException: Cannot instantiate class:
org.apache.activemq.jndi.ActiveMQInitialContextFactory
[Root exception is java.lang.ClassNotFoundException:
org.apache.activemq.jndi.ActiveMQInitialContextFactory]
19 Nov 2015 18:16:39.759 [22294] ERROR #splapptrc,J[108],P[118],
```

```
ImportedTransactionSet_AKRT.TransactionSet_JMS_AKRT,spl_javaop M[JavaOp.cpp:
trace:82] - javax.naming.spi.NamingManager.getInitialContext(NamingManager.java:
685)
19 Nov 2015 18:16:39.759 [22294] ERROR #splapptrc,J[108],P[118],
ImportedTransactionSet_AKRT.TransactionSet_JMS_AKRT,spl_javaop M[JavaOp.cpp:
trace:82] - javax.naming.InitialContext.getDefaultInitCtx(InitialContext.java:
318)
19 Nov 2015 18:16:39.759 [22294] ERROR #splapptrc,J[108],P[118],
ImportedTransactionSet_AKRT.TransactionSet_JMS_AKRT,spl_javaop M[JavaOp.cpp:
trace:82] - javax.naming.InitialContext.init(InitialContext.java:253)
19 Nov 2015 18:16:39.760 [22294] ERROR #splapptrc,J[108],P[118],
ImportedTransactionSet_AKRT.TransactionSet_JMS_AKRT,spl_javaop M[JavaOp.cpp:
trace:82] - javax.naming.InitialContext.<init>(InitialContext.java:227)
19 Nov 2015 18:16:39.760 [22294] ERROR #splapptrc,J[108],P[118],
ImportedTransactionSet_AKRT.TransactionSet_JMS_AKRT,spl_javaop M[JavaOp.cpp:
trace:82] - com.ibm.streamsx.messaging.jms.JMSConnectionHelper.
createAdministeredObjects s(JMSConnectionHelper.java:192)
19 Nov 2015 18:16:39.760 [22294] ERROR #splapptrc,J[108],P[118],
ImportedTransactionSet_AKRT.TransactionSet_JMS_AKRT,spl_javaop M[JavaOp.cpp:
trace:82] - com.ibm.streamsx.messaging.jms.JMSSource.initialize(JMSSource.java:
340) 19 Nov 2015 18:16:39.761 [22294] ERROR #splapptrc,J[108],P[118],
ImportedTransactionSet_AKRT.TransactionSet_JMS_AKRT,spl_javaop M[JavaOp.cpp:
trace:82] - com.ibm.streams.operator.internal.runtime.api.OpAdapter.
operatorInitialize (OpAdapter.java:92)
19 Nov 2015 18:16:39.761 [22294] ERROR #splapptrc,J[108],P[118],
ImportedTransactionSet_AKRT.TransactionSet_JMS_AKRT,spl_javaop M[JavaOp.cpp:
trace:82] -
```

Resolving the problem

Edit the Streams `.bashrc` file to reference the queue server client library, as described in the *IBM Opportunity Detect Administrator's Guide* in the topic titled "Installing MQ client libraries."

Deployment process failure

Symptoms

A deployment process fails to complete successfully.

The following error messages appear in the Streams log.

```
CDISR2517E The stdin mode is not the standard input for prompting for a
password. The tty mode is the standard input. The password prompt was
disabled.

CDISC5102E An error occurred while the system logged in to the instance.
See the previous error messages.
```

When you synchronize Streams by clicking **Synchronize with changed Streams configuration** on the **Settings & Detect Settings > Server Group** page, you see this warning: "Connection refused."

Causes

The most likely cause is incorrect or missed execution of the Streams First Steps application.

Resolving the problem

1. Log in to the Streams runtime server as the streamsadmin user.
2. Execute the following command:

   ```
   streamstool genkey
   ```

3. Restart the Streams instance and redeploy the workspace.

Workspace using queues fail to deploy with CDISP0467W error

Symptoms

A workspace containing a Queue connector fails to deploy. The trace log contains the following error.

```
com.ibm.unica.engine/UnicaEngine.spl:1938:1: CDISP0467W WARNING:
        The SaveStateForAllContainers function does not need to be declared stateful
        because it does not call any stateful functions.
Exception in thread "main" java.lang.UnsupportedClassVersionError:
        JVMCFRE003 bad major ver-sion; class=com/ibm/streamsx/messaging/jms/
        JMSSource, offset=6
at java.lang.ClassLoader.defineClass(ClassLoader.java:275)
at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:74)
at java.net.URLClassLoader.defineClass(URLClassLoader.java:540)
at java.net.URLClassLoader.defineClass(URLClassLoader.java:451)
at java.net.URLClassLoader.access$300(URLClassLoader.java:79)
at java.net.URLClassLoader$ClassFinder.run(URLClassLoader.java:1038)
at java.security.AccessController.doPrivileged(AccessController.java:310)
at java.net.URLClassLoader.findClass(URLClassLoader.java:429)
at java.lang.ClassLoader.loadClass(ClassLoader.java:660)
at java.lang.ClassLoader.loadClass(ClassLoader.java:626)
at com.ibm.streams.operator.internal.runtime.api.OperatorAdapter.loadOperatorClass
        (OperatorAdapter.java:386)
at com.ibm.streams.operator.internal.compile.OperatorCompileTime.
        verifyClassNameAndLibraries (OperatorCompileTime.java:175)
at com.ibm.streams.operator.internal.compile.OperatorCompileTime.
        executeCompileTimeChecks (OperatorCompileTime.java:137)
at com.ibm.streams.operator.internal.compile.
        OperatorCompileTime.main (OperatorCompileTime.java:108)
com.ibm.unica.qcio/QueueInputOutput.spl:150:3: CDISP9164E ERROR:
```

Resolving the problem

Set the JAVA_HOME variable as described in the *IBM Opportunity Detect and Interact Advanced Patterns Installation Guide*, in the topic titled "Configuring IBM InfoSphere Streams for use with Opportunity Detect."

Failed to get most recent runs for the workspace

Symptoms

After logging in to in to Opportunity Detect, when you select a workspace you see a popup window stating "Failed to get most recent runs for the workspace".

Causes

The following issues are the most common causes of this issue.

- Opportunity Detect design time could not connect to Opportunity Detect runtime (the Streams server). This is the most common cause of the problem.
- The database server is down.
- A network problem is affecting connections.

Resolving the problem

If the Streams server is down, start its services using the procedure in the *IBM Opportunity Detect and Interact Advanced Patterns Installation Guide* titled "Starting the Opportunity Detect Run Time server and Interact service."

If the database server is down, restart it.

For network problems, contact your system administrator.

Apache HttpClient error during Streams synchronization

Symptoms

When you synchronize Streams by clicking **Synchronize with changed Streams configuration** on the **Settings & Detect Settings > Server Group** page, you see a popup that states "org.apache.commons.httpclient.HttpClient incompatible with org.apache.commons.httpclient.HttpClient"

Causes

One or both of the Marketing Platform and Opportunity Detect deployments are not configured properly in the web application server.

Resolving the problem

1. Stop the following services:
   - StreamsRCS
   - RealTimeConnector
   - InteractService
2. In WebSphere, stop your Marketing Platform and Opportunity Detect deployments.
3. Follow the instructions in the *IBM Opportunity Detect and Interact Advanced Patterns Installation Guide* to set **Class loading and update detection** and **Manage Modules** properties for both Marketing Platformand Opportunity Detect.

   See Chapter 6: Deploying Opportunity Detect.
4. In WebSphere, restart your Marketing Platform and Opportunity Detect deployments.
5. Restart the StreamsRCS, RealTimeConnector, and InteractService.

Attempt to save a workspace or database connection fails

Symptoms

When you try to save a workspace or database connection, a popup warns that the process failed.

Resolving the problem

In WebSphere, under Security Settings for each node in your environment, set the **Component-managed authentication alias** and **Mapping-configuration alias**fields as described in the *IBM Opportunity Detect Installation Guide*, in the topic titled "Creating data sources in the web application server for Opportunity Detect."

Workspace fails to update

Symptoms

During workspace creation, you see the following error message: "Failed to update workspace." The trace.log has the following entry:

```
[WebContainer : 8] ERROR com.ibm.unica.detect.ui.service.WorkspaceService -
createOrUpdateWorkspace() caught exception for workspaceId = 'null'
and workspaceName = 'GS'
```

Resolving the problem

1. Clear the application server cache as described in the *IBM Opportunity Detect and Interact Advanced Patterns Installation Guide*, in the topic titled "Clearing the application server cache."

2. Check **Class loading and update detection** and **Manage modules** settings as described in the *IBM Opportunity Detect and Interact Advanced Patterns Installation Guide*, in the chapter titled "Deploying Opportunity Detect."

Login fails in a distributed environment

Symptoms

When a user logs in to Opportunity Detect, the user is redirected to the login screen again.

Resolving the problem

Ensure that the system time on all machines used in the distributed environment is the same.

Database access error in Streams log

Symptoms

In an installation that uses DB2 for system tables, after a workspace is deployed, you cannot start the deployment configuration. Some Streams jobs are unhealthy.

An error such as the following appears in the Streams log:

```
/home/streamsadmin/OpDetection/toolkits/unica_db_access/impl/include/
        UnicaDatabaseAccess.h: At global scope:
/home/streamsadmin/OpDetection/toolkits/unica_db_access/impl/include/
        UnicaDatabaseAccess.
/home/streamsadmin/OpDetection/toolkits/unica_db_access/impl/include/
        UnicaDatabaseAccess.h:44: error: 'SQLHENV' does not name a type
/home/streamsadmin/OpDetection/toolkits/unica_db_access/impl/include/
        UnicaDatabaseAccess.h:45: error: 'SQLHDBC' does not name a type
/home/streamsadmin/OpDetection/toolkits/unica_db_access/impl/include/
        UnicaDatabaseAccess.h:53: error: 'SQLHSTMT' does not name a type
/home/streamsadmin/OpDetection/toolkits/unica_db_access/impl/include/
        UnicaDatabaseAccess.h:69: error: 'SQLINTEGER' does not name a type
make: *** [build/function/com/ibm/unica/engine/LoadState.o] Error 1
```

Resolving the problem

1. If you are using DB2 for your system table database, ensure that the `include` directory is present under the DB2 client installation directory on the Streams server.

   If the `include` directory is not present, contact your database administrator.
2. Ensure that all `.h` (header) files are present in the `include` directory under the DB2 client installation directory on the Streams server.
3. Ensure that an accurate path is specified in the Streams `function.xml` file for the include and lib64 directories.

   The `function.xml` file is located in the `/home/streamsadmin/OpDetection/toolkits/unica_db_access/db.access/native.function` directory.

   The path should reflect the location of your DB2 database client. This is described in the *IBM Opportunity Detect and Interact Advanced Patterns Installation Guide*, in the topic titled "Editing Streams and Opportunity Detect XML files."

Oracle driver error

Symptoms

Opportunity Detect is installed using the manual database setup option, and the system tables are in Oracle. The following error occurs in the trace log.

```
org.apache.axis2.AxisFault: A connection could not be obtained for driver
class "oracle.jdbc.OracleDriver" and URL "jdbc:oracle:thin:@//[DB
Host]:[Port]/[Schema]".  You may have specified an invalid URL.
```

Resolving the problem

Take the following steps.

1. Add the `ojdbc7.jar` file on the runtime server in the following location.

   `/home/streamsadmin/OpDetection/StreamsRemoteControlService/wlp/usr/servers/StreamsRCS/dropins/StreamsRCS/WEB-INF/lib/`
2. Restart the Streams instance and the StreamsRCS service.

Issues configuring ODBC.ini and executing ISQL on the Streams server

Symptoms

When you use Oracle for your system tables or profile data, iSQL fails to run.

Resolving the problem

Use a fully qualified name for the database server in the `ODBC.ini` file.

After applying a Fix Pack, new features are not present

Symptoms

After you upgrade by applying a Fix Pack, you do not have the expected new features or functionality.

Resolving the problem

Clear the application server cache as described in the *IBM Opportunity Detect and Interact Advanced Patterns Installation Guide*, in the topic titled "Clearing the application server cache."

Monitoring - InstanceAlreadyExistsException

Symptoms

An InstanceAlreadyExists exception occurs in the monitoring tool

Causes

This error may be due to duplicate deployment configuration names.

Resolving the problem

The log provides the name of deployment configuration. To resolve the issue, modify duplicate deployment configuration name and deploy the workspace again.

All deployments show the same number of total transactions in the monitoring utility

Symptoms

After a Streams restart, the monitoring utility shows the same number of total transactions for all deployments.

Resolving the problem

This issue occurs when Streams has not been shut down gracefully. To avoid the problem, shut down Streams as follows.

1. In the Opportunity Detect user interface, stop all running deployments.

2. On the Streams server, manually remove all job ID files located in the `apps/RealTimeProxy` directory under your Opportunity Detect installation.

   The format of these files is rtp.jobid.*Streams_instance_name*@streamsadmin.*host*
   For example, `rtp.jobid.streams1@streamsadmin.examplehost`

InfoSphere Streams installation fails

When IBM Opportunity Detect is installed as root, InfoSphere Streams is installed by using the streamsadmin user. InfoSphere Streams installation by using streamsadmin might fail if SELinux (Secure Enhanced Linux)is enabled.

To verify if SELinux is installed, run the `command/usr/sbin/sestatus` command.

Disabled SELinux by setting `SELINUX=disabled in /etc/selinux/config`.

Appendix. Opportunity Detect and Interact Advanced Patterns configuration properties

This section describes the Opportunity Detect and Interact Advanced Patterns configuration properties on the Configuration page.

IBM Opportunity Detect and Interact Advanced Patterns | Navigation

Properties in this category specify values that are used internally to navigate among IBM Marketing Software products.

welcomePageURI

Description

The Uniform Resource Identifier of the IBM Opportunity Detect index page. This value is used internally by IBM Marketing Software applications. Changes to this value are not recommended.

Default value

`/index.jsp`

seedName

Description

Used internally by IBM Marketing Software applications. Changes to this value are not recommended.

Default value

`Detect`

type

Description

Used internally by IBM Marketing Software applications. Changes to this value are not recommended.

Default value

`Detect`

httpPort

Description

The port number that is used by the application server for connections to the Opportunity Detect application.

Default value

`7001`

httpsPort

Description

The port number that is used by the application server for secure connections to the Opportunity Detect application.

Default value

7001

serverURL

Description

The URL of the Opportunity Detect installation. Accepts either the HTTP or HTTPS protocol. If you are on a clustered environment and choose to use ports that are different from the default ports 80 or 443 for your deployment, do not use a port number in the value of this property.

If users access Opportunity Detect with the Chrome browser, use the fully qualified domain name (FQDN) in the URL. If the FQDN is not used, the Chrome browser cannot access the product URLs.

Important: If IBM Marketing Software products are installed in a distributed environment, you must use the machine name rather than an IP address in the navigation URL for all of the applications in the suite.

Default value

[server-url]

logoutURL

Description

Used internally. Changes to this value are not recommended.

IBM Marketing Platform uses this value to call the logout handler of each registered application if the user clicks the logout link in IBM Marketing Software.

serverURLInternal

Description

Used internally. Changes to this value are not recommended.

displayName

Description

Used internally. Changes to this value are not recommended.

Default value

Opportunity Detect

IBM Opportunity Detect and Interact Advanced Patterns | System | Streams Remote Control Web Service

The property in this category specifies the URL for the IBM InfoSphere Streams remote control web service. Opportunity Detect Design Time communicates with Opportunity Detect Run Time over this service.

ServerURL

Description

The person who installs the product sets this property value during installation. The default port number is 8080.

Default value

`http://[SRCSHost]:[SRCSPort]/axis2/services/RemoteControl`

IBM Opportunity Detect and Interact Advanced Patterns | System | Real Time Connector

The property in this category specifies the URL for the web service used when Interact is integrated with Interact Advanced Patterns or when the Web Service connector is used for input data.

ServerURL

Description

The person who installs the product sets this property value during installation. The default port number is 8282.

Default value

`http://[RealTimeConnectorHost]:[RealTimeConnectorPort]/servlets/StreamServlet`

IBM Opportunity Detect and Interact Advanced Patterns | System | Monitoring

Properties in this category specify values that affect the monitoring tool.

Poll Interval (In Seconds)

Description

The number of seconds that the monitoring service waits between two successive polls of the Streams server for the statistics. The default is 300 seconds, or 5 minutes.

Default value

`300`

Retaining Time (In Days)

Description

The number of days the monitoring service should keep the polled data in the database. The default is 10 days. Data that is older than the time specified here is purged.

Default value

`10`

Related concepts:

"Monitoring tool configuration" on page 79

IBM Opportunity Detect and Interact Advanced Patterns | System | Processing Options

Properties in this category specify values that affect the monitoring tool.

Cache profile records

Description

Opportunity Detect can cache profile data, which provides optimal performance. To enable caching of profile data, set the value of this property to `True`.

If you have very large profile data sets, you might want to retain the default value of this property, which is `False`. This disables caching of profile data and eliminates the out of memory issues that caching large amounts of profile data can cause.

If you change this property value, you must restart your web application server, the Streams instance, and the StreamsRCS service, and redeploy all affected deployments.

Default value

`False`

IBM Opportunity Detect and Interact Advanced Patterns | logging

The property in this category specifies the location of the Opportunity Detect log file.

log4jConfig

Description

The location of the configuration file that Opportunity Detect uses for logging. This value is set automatically during installation, but if you change this path, you must restart the web application server to apply the change.

Default value

`[absolute-path]/conf/detect_log4j.properties`

IBM Interact Advanced Patterns | System | Interact Design Service

The property in this category specifies the URL for the web service that allows Interact to automatically create and deploy advanced patterns when Interact is integrated with Interact Advanced Patterns.

ServerURL

Description

This web service is the integration point between Interact and Interact Advanced Patterns design time. The person who installs the product sets this property value during installation. The default port number is 8181.

Default value

`http://[InteractServiceHost]:[InteractServicePort]/axis2/services/InteractDesignService`

Properties for configuring email notifications

These configuration properties must be set if you want to enable email notifications for system alerts and user notifications.

IBM Marketing Software | General | Communication | Email

Properties in this category are used to configure the Marketing Platform to send emails to users for system alerts and notifications.

Enable email communication

Description

When set to `True`, the Marketing Platform attempts to send emails to users for system alerts and notifications. The other properties in this category must also be set to enable this feature.

Default value

`False`

Email server protocol

Description

Specifies the protocol on the mail server that is used for sending system alerts and notifications to users. This is required for email notifications.

Default value

`smtp`

Email server host

Description

Specifies the name of the mail server used for sending system alerts and notifications to users. This is required for email notifications.

Default value

`localhost`

Email server port

Description

Specifies the port of the mail server used for sending system alerts and notifications to users. This is required for email notifications.

Default value

`25`

'From' address for emails

Description

Specifies the account from which system alert and notification emails are sent. If authentication is required on your mail server, use the email address of the account that you used when you saved a mail server account name and password as a data source in a Marketing Platform user account. This is required for email notifications.

Default value

Not defined

Authentication required for mail server?

Description

Specifies whether the mail server requires authentication.

Default value

`False`

IBM Marketing Software user for email account

Description

Specifies the user name of the Marketing Platform account where the email credentials are stored as a data source.

Required for notifications, only if your mail server requires authentication.

Default value

`asm_admin`

Data source for email account

Description

Specifies the name of the data source in the Marketing Platform account where the email credentials are stored.

Required for notifications, only if your mail server requires authentication.

Default value

`emailDS`

Related tasks:

"Configuring email notifications in IBM Marketing Software" on page 85

IBM Marketing Platform | Notifications

Properties in this category control system behavior for notifications that IBM Marketing Software products can send to users.

How many days to retain alerts

Description

Specifies the amount of time, in days, that a system alert is retained in the system for historical purpose after the expiry date, which is provided by the application that sent the alert. Alerts older than the specified number of days are deleted from the system.

Default value

`90`

How frequently to send emails (in minutes)

Description

Specifies how many minutes the system waits before sending any new notification emails.

Default value

30

Maximum re-tries for sending email

Description

Specifies how many times the system attempts to send notification emails when an initial attempt to send fails.

Default value

1

Related tasks:

"Configuring email notifications in IBM Marketing Software" on page 85

Before you contact IBM technical support

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

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

Note: Technical Support does not write or create API scripts. For assistance in implementing our API offerings, contact IBM Professional Services.

Information to gather

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

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

System information

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

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

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

Contact information for IBM technical support

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

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

Notices

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

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

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

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

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

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

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

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

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

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

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

IBM Corporation
B1WA LKG1
550 King Street
Littleton, MA 01460-1250
U.S.A.

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

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

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

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

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

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

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

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating

platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

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

Trademarks

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

Privacy Policy and Terms of Use Considerations

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

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

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

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

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

Printed in USA