Oracle Financial Services

Currency Transaction Reporting

Administration Guide

Release 8.0.2.0.0

October 2019

Oracle Financial Services Currency Transaction Reporting **Administration Guide**

Release 8.0.2.0.0
October 2019

Part Number: E60570_20

Oracle Financial Services Software, Inc.
1900 Oracle Way
Reston, VA 20190

Document Number: E60570_20
Fifth Edition (October 2019)

Oracle Financial Services Software, Inc.
1900 Oracle Way
Reston, VA 20190
Phone: 703-478-9000
Fax: 703-318-6240
Internet: www.oracle.com/financialservices

Revision History

Table 1 describes the revision history of *Oracle Financial Services Currency Transaction Reporting Administration Guide*.

Table 1. Revision History

Date	**Edition**	**Description**
October 2019	Fifth Edition	In Chapter 2, *Security Configuration*, updated the following sections: ● *Configuring Exempt Filer Bank* ● *Configuring Affiliated Bank*
December 2017	Fourth Edition	In Chapter 7, *Administrative Utilities*, added the following sections: ● *Configuring the Exemption Password*
February 2016	Third Edition	The third release of *Oracle Financial Services Currency Transaction Reporting 8.0.2 Administration Guide.*
June 2015	Second Edition	The second release of *Oracle Financial Services Currency Transaction Reporting 1.2.1 Administration Guide.*
September 2012	First edition.	The first release of *Oracle Financial Services Currency Transaction Reporting 1.2 Administration Guide.*

Contents

List of Tables

List of Figures

About this Guide

This guide explains the concepts behind the Oracle Financial Services Financial Services Currency Transaction Reporting, and provides comprehensive instructions for proper system administration, as well as daily operations and maintenance.

Disclaimer: In this guide, Oracle Financial Services Currency Transaction Reporting and Oracle Financial Services Behavior Detection Platform are used interchangeably. As an Administrator, you can configure and manage both Oracle Financial Services Currency Transaction Reporting and Oracle Financial Services Behavior Detection Platform systems.

This section focuses on the following topics:

- Who Should Use this Guide
- Scope of this Guide
- How this Guide is Organized
- Where to Find More Information
- Conventions Used in this Guide

Who Should Use this Guide

The *Oracle Financial Services Currency Transaction Reporting Administration Guide* is designed for use by Oracle Financial Services Installers and System Administrators. Their roles and responsibilities, as they operate within Oracle Financial Services Currency Transaction Reporting and Oracle Financial Services Behavior Detection, include the following:

- **Oracle Financial Services Installer:** This user installs and configures the Oracle Financial Services Applications and the client-specific solution sets at a deployment site. This user also installs upgrades, and additional solution sets. It requires access to deployment-specific configuration information (For example, machine names, and port numbers).
- **System Administrator:** This user configures, maintains, and adjusts the system and is usually an employee of a specific Oracle client. The System Administrator maintains user accounts and roles, archives data, and loads data feeds.

Scope of this Guide

This guide provides step-by-step instructions for the user who configures, maintains, and adjusts the system.

How this Guide is Organized

The *Oracle Financial Services Currency Transaction Reporting Administration Guide* includes the following chapters:

- Chapter 1, *Oracle Financial Services Behavior Detection,* provides a brief overview of Oracle Financial Services Behavior Detection and its components.
- Chapter 2, *Behavior Detection Jobs,* provides an overview of the Oracle Financial Services Job Protocol and procedures for performing various tasks that relate to starting, stopping, and recovering jobs.
- Chapter 3, *Security Configuration,* covers the required day-to-day operations and maintenance of users, groups, and organizational units.
- Chapter 4, *Data Ingestion,* describes the operation and process flow of Data Ingestion subsystem components.
- Chapter 5, *Post-Processing Tasks,* describes the derivation and aggregation of data through workflows after the data ingestion process completes.
- Chapter 6, *Batch Processing Utilities,* explains how to customize features that affect presentation of user information on the desktop.
- Chapter 7, *Administrative Utilities,* provides information about the Oracle Financial Services database utilities related to the batch process.
- Chapter 8, *CTR Batch Execution,* provides information about about the CTR Batch process.
- Appendix A, *Logging*,, describes the Oracle Financial Services logging feature.

Where to Find More Information

For more information about Oracle Financial Services Currency Transaction Reporting, see the following documents, which can be found at

http://docs.oracle.com/cd/E60570_01/homepage.htm:

- *Oracle Financial Services Behavior Detection Platform Installation Guide*
- *Oracle Financial Services Currency Transaction Reporting Installation Guide*
- *Oracle Financial Services Currency Transaction Reporting Data Interface Specification Guide*
- *Oracle Financial Services Currency Transaction Reporting Configuration Guide*
- *Oracle Financial Services Currency Transaction Reporting User Guide*
- *Oracle Financial Services Currency Transaction Reporting Release Notes*
- *Oracle Financial Services Analytical Applications Infrastructure Installation and Configuration Guide*
- *Oracle Financial Services Analytical Applications Infrastructure User Manual*

For installation and configuration information about Sun Java System, BEA, and Apache software, see the appropriate documentation that is available on the associated web sites.

Conventions Used in this Guide

Table 2 lists the conventions used in this guide.

Table 2. Conventions Used in this Guide

This convention...	Stands for...
Italics	● Names of books, chapters, and sections as references ● Emphasis
Bold	● Object of an action (menu names, field names, options, button names) in a step-by-step procedure ● Commands typed at a prompt ● User input
`Monospace`	● Directories and subdirectories ● File names and extensions ● Process names ● Code sample, including keywords and variables within text and as separate paragraphs, and user-defined program elements within text
<Variable>	● Substitute input value

CHAPTER 1

Oracle Financial Services Behavior Detection

This chapter provides a brief overview of the Oracle Financial Services Behavior Detection Platform architecture and operations. It also includes new features for this release. This chapter focuses on the following topics:

- Technology Compatibility
- About the Oracle Financial Services Architecture
- About Oracle Financial Services Operations

Technology Compatibility

Oracle Financial Services is able to meet the environmental needs of its customers by providing support for third-party tools such as WebSphere, WebLogic, Linux, Oracle, and Informatica. See the *Installation Guide* for more information about these tools.

About the Oracle Financial Services Architecture

An architecture is a blueprint of all the parts that together define the system: its structure, interfaces, and communication mechanisms. A set of functional views can describe an architecture.

The following views illustrate the implementation details of the application's architecture:

- **Component View:** Illustrates system components and their dependencies.
- **Security View:** Emphasizes the security options between processing nodes through a specialized deployment view.

The following sections describe these views.

Component View

This view describes the concept that a series of tiers and subsystems compose the Oracle Financial Services architecture. Each tier can contain all subsystems. Subsystems, in turn, include one or more components that are divided into small installable units. A solution set requires installation of the associated Oracle Financial Services architecture.

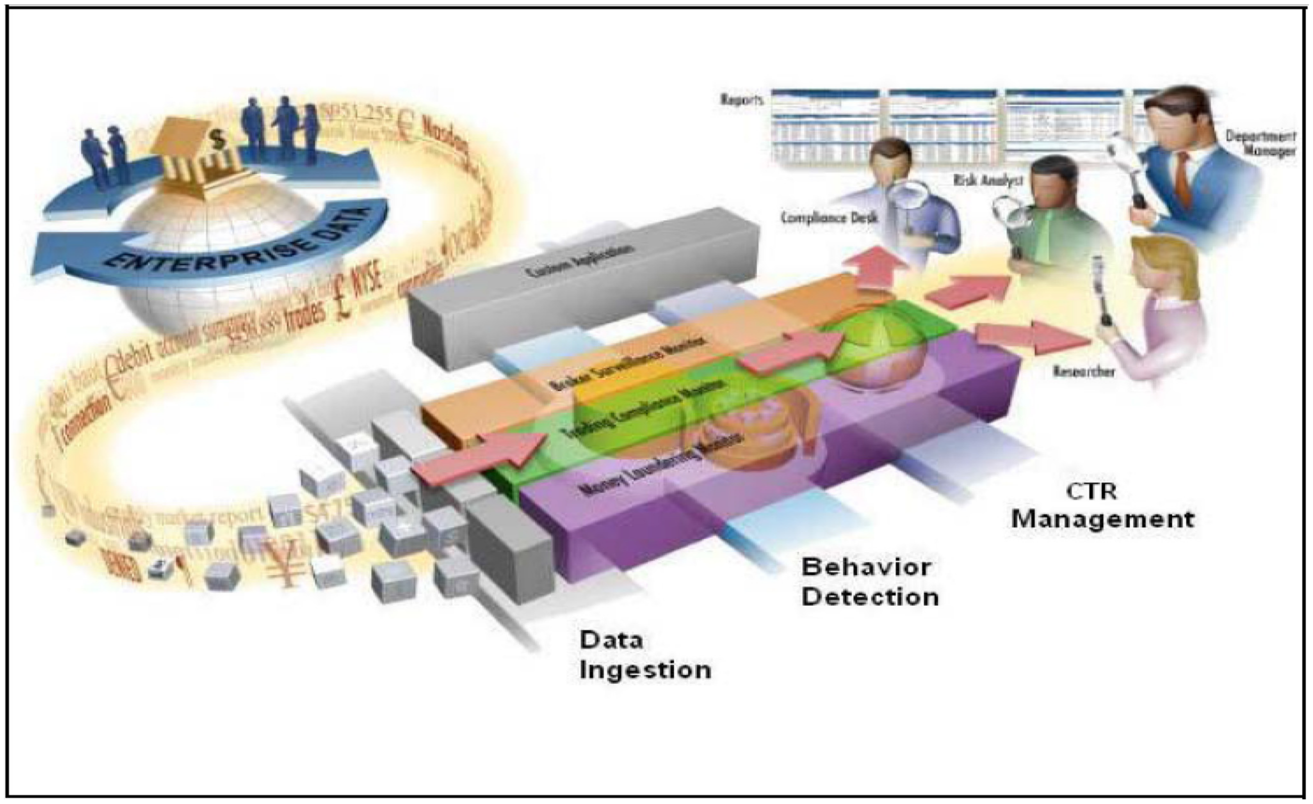


Figure 1. Oracle Financial Services Architecture - Overview

Each tier can contain all subsystems. Subsystems, in turn, include one or more components that are divided into small installable units. A solution set requires installation of the associated Oracle Financial Services components.

Tiers

The Currency Transaction Reporting solution has two tiers:

- The **Behavior Detection Platform** defines a foundation for building Currency Transaction Reporting solution sets. It provides core data mining services, frameworks, and tools. The application also includes interface packages that abstract non-standard or proprietary commercial off-the-shelf (COTS) products. Deployment of multiple Currency Transaction Reporting solution sets can occur on a single installation.
- Each **Oracle Financial Services solution set** (CTR, Fraud Detection and Anti-Money Laundering) extends the Oracle Financial Services framework. Each adds domain-specific content to provide the required services for addressing a specific business problem. It includes reusable domain artifacts such as scenarios, input data transformation code, and profiling scripts. A solution set also provides the required presentation packages and custom application objects for supporting user-interface functionality specific to the business domain.

Subsystems

The application is composed of the following four subsystems:

- **Data Ingestion:** Provides data preparation logical functions, which include adapters for files and messages. The functions also include mappings for data derivations and aggregations.
- **Behavior Detection:** Provides data access, behavior detection, and job services, which include the Financial Services Data Model (FSDM) and scenarios specific to a particular solution set.

A set of components further divides each subsystem. Components are units of a subsystem that can be installed separately onto a different server. Table 3 outlines the definition for the subsystems and components. In some cases, however, individual deployments can add subsystems or components to meet a client's custom requirements.

Table 3. Subsystems and their Components

Common Name	Directory Name	Contents
Data Ingestion	bdf	Java components, Informatica components, scripts, and stored procedures
Database Tools	database/db_tools	For DB tools directory
Detection Algorithms	algorithms	C++ behavior detection algorithms
Scenario Manager	toolkit	Job and scenario editors
Financial Services Data Model	database	Database utilities and database creation scripts

Security View

The security view of the architecture and use of security features of the network in a Behavior Detection architecture deployment is illustrated in Figure 2. Behavior Detection uses inbuilt SMS for its authentication and authorization. The SMS has a set of database tables which store information about user authentication.

Installation of 128-bit encryption support from Microsoft can secure the Web browser. Oracle encourages using the Secure Socket Layer (SSL) between the Web browser and Web server for login transaction, While the Web Application server uses a browser cookie to track a user's session, this cookie is temporary and resides only in browser memory. When the user closes the browser, the system deletes the cookie automatically.

The application uses Advanced Encryption Standard (AES) security to encrypt passwords that reside in database tables in the configuration schema on the database server and also encrypts the passwords that reside in configuration files on the server.

.

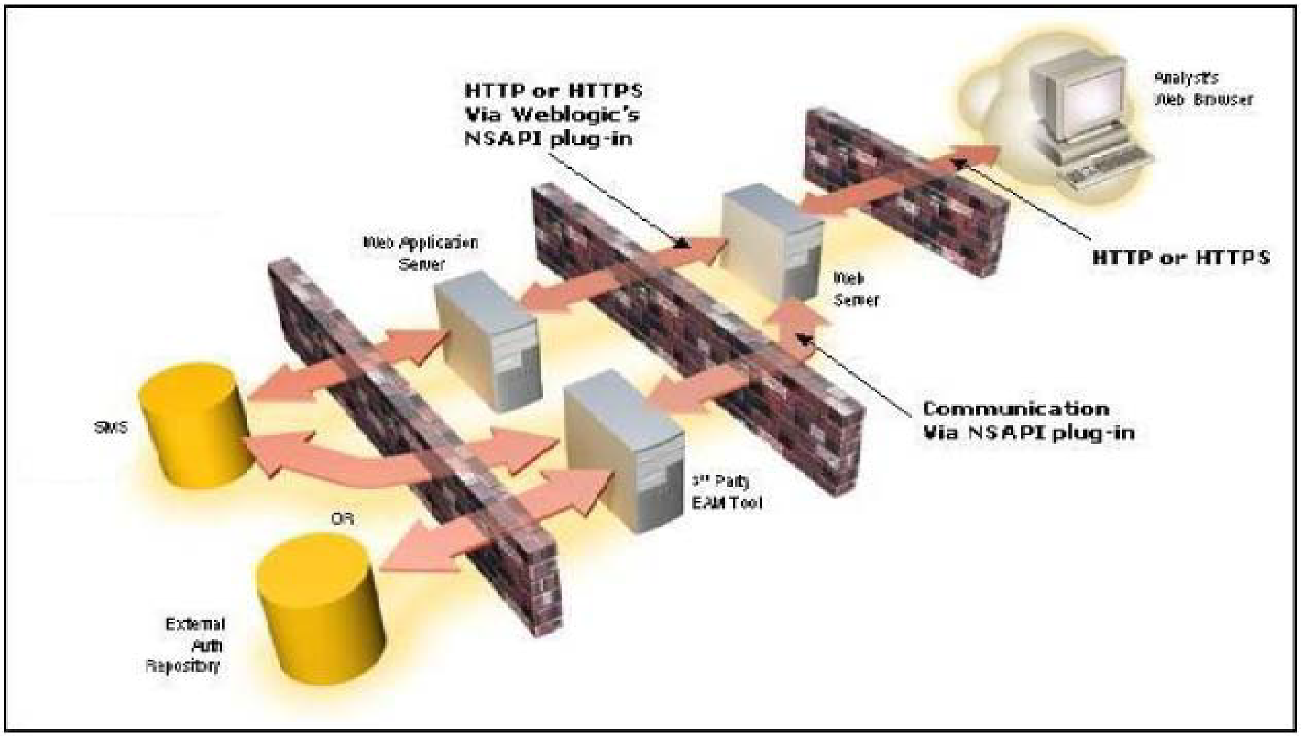


Figure 2. Oracle Financial Services/ Architecture—Security View

The EAM tool is an optional, third-party, pluggable component of the security view. The tool's integration boundaries provide an Authorization header, form field with principal, or embedded principal to the Web Application server through a Web server plug-in. The tool also passes the same user IDs that the Currency Transaction Reporting directory server uses.

About Oracle Financial Services Operations

As the Oracle administrator, you can coordinate the overall operations of the application Data Ingestion, Behavior Detection, and Post-Processing.

In a production environment, an Oracle client typically establishes a processing cycle to identify occurrences of behaviors of interest (that is, scenarios) on a regular basis.

Each cycle of the Currency Transaction Reporting process begins with Data Ingestion, Behavior Detection, and Post-Processing, which prepares the detection results for presentation for the users.

Several factors determine specific scheduling of these processing cycles, including availability of data and the nature of the behavior that the system is to detect. The following sections describe each of the major steps in a typical production processing cycle:

- Start Batch
- Data Ingestion
- Behavior Detection
- Post-Processing

- End Batch

Start Batch

Using the Batch Control Utility, you can manage the beginning of a batch process (See *Chapter 7 - Administrative Utilities*, for more information).

Data Ingestion

Your raw business data can be loaded into the Oracle Financial Services Data Model (FSDM) in various ways. The following approaches are available either through the OFSDF Common Staging Area Model (CSA) or converting the raw data into Data Interface Specification (DIS) files.

The *Data Interface Specification (DIS)* contains a definition of the flat file data ingestion process for each solution set. For more information about data ingestion, see the *Oracle Financial Services Behavior Detection Administration Guide.*

Behavior Detection

During Behavior Detection, Detection Algorithms control the scenario detection process. The Detection Algorithms search for events and behaviors of interest in the ingested data. Upon identification of an event or behavior of interest, the algorithms record a match in the database.

The application executes the following processes in this order to find and record scenario matches:

1. The system populates temporary tables in the database; some scenarios depend on these tables for performance reasons.
2. A network creation process generates and characterizes networks, filtering the links that the system evaluates in the construction of these networks.
3. A match creation process creates matches based on detection of specific sequences of events in data that correspond to patterns or the occurrences of prespecified conditions in business data. The process also records additional data that the analysis of each match may require.

Post-Processing

During Post-Processing, the detection results are prepared for presentation to users. This preparation is dependent upon the following processes:

1. An alert creation process packages the scenario matches as units of work (that is, alerts), potentially grouping similar matches together, for disposition by end users (See section *To Run Match Alert Creator* for more information).
2. During batch execution, alerts are converted into CTR records. See *Batch Execution of CTR*, for more information.

End Batch

The system ends batch processing when processing of data from the Oracle client is complete (See section *Ending a Batch Process*, for more information).

CHAPTER 2

Behavior Detection Jobs

This chapter provides an overview of the Oracle Financial Services Job Protocol and then explains how the System Administrator monitors jobs, and starts and stops jobs when necessary. In addition, it describes the necessary scripts that you use for jobs. This chapter focuses on the following topics:

- About the Oracle Financial Services Job Protocol
- Performing Dispatcher Tasks
- Performing Job Tasks
- Clearing Out the System Logs
- Recovering Jobs from a System Crash

Note: If Anti Money Laundering (AML) is not enabled, thresholds can be edited through the Scenario Manager.

About the Oracle Financial Services Job Protocol

The system initiates all jobs by using a standard operational protocol that utilizes each job's metadata, which resides in a standard set of database tables. Oracle Financial Services Job Protocol processes include the following:

- `dispatcher`: Polls the job metadata for new jobs that are ready for execution. This daemon process starts a `mantas` process for each new job.
- `mantas`: Creates a new job entry based on a template for the job that has the specific parameters for this execution of the job (that is, it clones a new job).

As an Oracle Financial Services administrator, you invoke the `dispatcher` and `mantas` processes by running the shell scripts in Table 4.

Table 4. Shell Scripts that Call *`mantas`* Processes

Process	Description
`start_mantas.sh`	Starts all jobs. This script invokes the **cloner** and `mantas` processes. This is the integration point for a third-party scheduling tool such as Maestro or AutoSys.
`start_chkdisp.sh`	Calls on the check_dispatch.sh script to ensure that the `dispatcher` runs.
`stop_chkdisp.sh`	Stops the `dispatcher` process.
`restart_mantas.sh`	Changes job status codes from the ERR status to the RES status so that the `dispatcher` can pick up the jobs with the RES status.
`recover_mantas.sh`	Changes job status codes for jobs that were running at the time of a system crash to the ERR status. After running this script, the `restart_mantas.sh` script must be run to change the ERR status code to RES in order for the `dispatcher` to be able to pick up these jobs.

In the Oracle Financial Services Job Protocol, the processes use a variety of metadata that the database provides. Some of this metadata specifies the jobs and their parameters that are associated with the regular operations of an installation. Some of this metadata captures the status of job execution and is useful for monitoring the progress of an operational cycle.

The following sections describe how the processes and metadata interact in the Oracle Financial Services Job Protocol.

Understanding the Oracle Financial Services Job Protocol

These templates associate an algorithm to run with parameters that the algorithm requires. Job Templates are grouped together to run in parallel through Job Template Groups in the `KDD_JOB_TEMPLATE` table. Template groups enable you to identify what jobs to run.

Table 5 provides an example of a job template group with two job templates.

Table 5. `KDD_JOB_TEMPLATE` with Sample Job Template Group

JOB_ID	TEMPLATE_GROUP_ID
37	1
41	1

Understanding the Dispatcher Process

The `dispatcher` process polls the job metadata waiting for jobs that need to be run. To control system load, the `dispatcher` also controls the number of jobs that run in parallel.

Generally, the dispatcher process should be running continuously, although it is possible to run jobs without a dispatcher.

For each job in the template group, the dispatcher runs a `mantas` process. The `dispatcher` tracks jobs for status and completion, and reports any failure to the dispatch log.

See *Starting the Dispatcher*, on page 11, and *Stopping the Dispatcher*, on page 11, for more information.

Understanding the mantas Process

The `dispatcher` runs jobs using the `mantas` process. This process runs the appropriate algorithm, tracks status in the `KDD_JOB` and `KDD_RUN` tables. One `mantas` process can result in multiple `KDD_RUN` records.

The `mantas` process also logs job progress and final status.

Applying a Dataset Override

You use the dataset override feature to permit dataset customizations specific to your site, which can be retained outside of the scenario metadata. The override to a dataset definition is stored in a file accessible by the Behavior Detection engine. The dataset override feature allows improved performance tuning and the ability to add filters that are applicable only to your site's dataset.

When the system runs a job, it retrieves the dataset definition from the database. The Behavior Detection engine looks in the configured directory to locate the defined dataset override. The engine uses the override copy of the dataset instead of the copy stored in the scenario definition in the database, if a dataset override is specified.

The following constraints apply to overriding a dataset:

- The columns returned by the dataset override must be identical to those returned by the product dataset. Therefore, the dataset override does not support returning different columns for a pattern customization to use.
- The dataset override can use fewer thresholds than the product dataset, but cannot have more thresholds than the product dataset. Only thresholds applied in the dataset from the scenario are applied.

If a dataset override is present for a particular dataset, the override applies to all jobs that use the dataset.

Configuring the Dataset Override Feature

The following section provides instructions to configure the directory for the Behavior Detection engine, for locating the defined dataset override.

To configure a dataset override, follow these steps:

1. Modify the `install.cfg` file for algorithms to identify the directory where override datasets are stored.

   The file resides in the following directory:

   ```
   <install_dir>/behavior_detection/algorithms/MTS/mantas_cfg/
   install.cfg
   ```

   The dataset override is specified with this property:

   ```
   kdd.custom.dataset.dir
   ```

   Note: Specify the directory using a full directory path, not a relative path. If you do not (or this property is not in the `install.cfg` file), the system disables the dataset overrides automatically.

2. Create the dataset override file in the specified directory with the following naming convention:

   ```
   dataset<DATASET_ID>.txt
   ```

   Note: The contents of the file should start with the SQL definition in `KDD_DATASET.SQL_TX`. This SQL must contain all of the thresholds still represented (for example, `@Min_Indiv_Trxn_Am`).

Performing Dispatcher Tasks

The **dispatcher** service runs on the server on which the application is installed. Once the dispatcher starts, it runs continuously unless a reason warrants shutting it down or it fails due to a problem.

This section describes the following:

- *Setting Environment Variables*
- *Starting the Dispatcher*
- *Stopping the Dispatcher*
- *Monitoring the Dispatcher*

Setting Environment Variables

Environment variables are set up during the installation process. These generally do not require modification thereafter.

All behavior detection scripts and processes use the `system.env` file to establish their environment.

About the *`system.env`* File

Table 6 describes environment variables in the `system.env` file.

Table 6. Environment Variables in system.env File

Variable	Description
`KDD_HOME`	Install path of the Oracle Financial Services software.
`KDD_PRODUCT_HOME`	Install path of the solution set. This is a directory under `KDD_HOME`.

Table 7 describes database environment variables in the `system.env` file.

Table 7. Database Environment Variables in system.env File

Variable	Environment	Description
`ORACLE_HOME`	Oracle	Identifies the base directory for the Oracle binaries. You must include: ● `$ORACLE_HOME` and `$ORACLE_HOME/bin` in the PATH environment variable value. ● `$ORACLE_HOME/lib` in the `LD_LIBRARY_PATH` environment variable value.
`ORACLE_SID`	Oracle	Identifies the default Oracle database ID/name to which the application connects.
`TNS_ADMIN`	Oracle	Identifies the directory for the Oracle network connectivity, typically specifying the connection information (SID, Host, Port) for accessing Oracle databases through SQL*NET.

Table 8 shows operating system variables in the `system.env` file.

Table 8. Operating System Environment Variables in system.env File

Variable	Description
PATH	Augmented to include `$KDD_HOME/bin` and the `$ORACLE_HOME, $ORACLE_HOME/bin` pair (for Oracle).
LD_LIBRARY_PATH, LIBPATH, SHLIB_PATH (based on operating system)	Augmented to include `$KDD_HOME/lib` and `$ORACLE_HOME/lib` (for Oracle)

Starting the Dispatcher

Although multiple jobs and `mantas` instances can run concurrently in the application, only one `dispatcher` service per database per installation should run at one time.

The application provides a script to *check* on the status of the `dispatcher` automatically and restart it, if necessary. Oracle recommends this method of running the `dispatcher`.

To start the `dispatcher`, follow these steps:

1. Verify that the dispatcher is not already running by typing
   `ps -ef | grep dispatch` and pressing **Enter** at the system prompt.

   If the dispatcher is running, an instance of the dispatcher appears on the screen for the server. If the dispatcher is not running, proceed to Step 2.

2. Type `start_chkdisp.sh <sleep time>` and press **Enter** at the system prompt to start the `dispatcher`.

   The dispatcher queries the database to check for any new jobs that need to be run. In between these checks, the dispatcher sleeps for the time that you specify through the `<sleep time>` parameter (in minutes).

   Optional parameters include the following:

- `dispatch name`: Provides a unique name for each `dispatcher` when running multiple `dispatchers` on one machine.
- `JVM size`: Indicates the amount of memory to allocate to Java processing.

Caution: For 32-bit Linux configurations, Oracle recommends running with the default JVM size (128 MB) due to 2 GB process limit.

The script executes and ends quickly. The dispatcher starts and continues to run in the background.

Stopping the Dispatcher

You do not normally shut down the dispatcher except for reasons such as the following:

- Problems while executing scenarios, make it necessary to stop processing.
- The dispatcher and job processes are reporting errors.
- The dispatcher is not performing as expected.
- You must shut down the system for scheduled maintenance.
- You want to run the `start_mantas.sh`, `restart_mantas.sh`, or `recover_mantas.sh` script without the dispatcher already running. You can then save your log files to the server on which you are working rather than the server running the dispatcher.

Caution: If you shut down the dispatcher, all active jobs shut down with errors.

When you are ready to restart the dispatcher and you want to see which jobs had real errors and which jobs generated errors only because they were shut down during processing, review the error messages in the job logs.

For those jobs that shut down and generate errors because the dispatcher shut down, a message similar to the following appears: `Received message from dispatcher to abort job`. If the job generates a real error, a message in the job log file indicates the nature of the problem.

To view active jobs and then shut down the dispatcher, follow these steps:

1. Type **`ps -efw | grep mantas`** and press **Enter** at the system prompt.

   All instances of the `mantas` process that are running appear on the screen. Only one instance of `mantas` should run for each active job.

2. Type **`stop_chkdisp.sh <`**dispatcher **`name>`** and press **Enter** at the system prompt.

   This script shuts down the dispatcher.

Monitoring the Dispatcher

The `install.cfg` file that was set up during server installation contains the `kdd.dispatch.joblogdir` property that points to a log file directory. The log directory is a repository that holds a time-stamped record of `dispatcher` and job processing events.

Each time the dispatcher starts or completes a job, it writes a status message to a file called `dispatch.log` in the log directory. This log also records any failed jobs and internal dispatcher errors. The `dispatch.log` file holds a time-stamped history of events for all jobs in the chronological sequence that each event occurred.

To monitor the `dispatch.log` file as it receives entries, follow these steps:

1. Change directories to the log directory.
2. Type **`tail -f dispatch.log`** and press **Enter** at the system prompt.

   The log file scrolls down the screen.

3. Press **Ctrl+C** to stop viewing the log file.
4. Type **`lpr dispatch.log`** and press **Enter** at the system prompt to print the `dispatch.log` file.

   Note: The `dispatch.log` file can be a lengthy printout.

Performing Job Tasks

At the system level, the Oracle administrator can start, restart, copy, stop, monitor, and diagnose jobs.

This section covers the following topics:

- Understanding the Job Status Codes
- Starting Jobs
- Starting Jobs without the Dispatcher
- Restarting a Job
- Restarting Jobs without the Dispatcher
- Stopping Jobs
- Monitoring and Diagnosing Jobs

Understanding the Job Status Codes

The following status codes are applicable to job processing and the dispatcher. The Oracle Financial Services administrator sets these codes through an Currency Transaction Reporting Job Editor:

- **NEW (start):** Indicates a new job that is ready to be processed.
- **RES (restart):** Indicates that restarting the existing job is necessary.
- **IGN (ignore):** Indicates that the dispatcher should ignore the job and not process it. This status identifies Job Templates.

The following status codes appear in the `KDD_JOB` table when a job is processing:

- **RUN (running):** Implies that the job is running.
- **FIN (finished):** Indicates that the job finished without errors.
- **ERR (error):** Implies that the job terminated due to an error.

Starting Jobs

The Oracle administrator starts jobs by running the `start_mantas.sh` script.

To start a new job, follow these steps:

1. Create the new job and job description through the Oracle Financial Services Job Editor.

   The application automatically assigns a unique ID to the job when it is created.

2. Associate the new job to a Job Template Group using the `KDD_JOB_TEMPLATE` table (See section *Understanding the Oracle Financial Services Job Protocol*, on page 8, for more information).
3. Execute the `start_mantas.sh` script as follows:

   ```
   start_mantas.sh <template ID>
   ```

The following events occur automatically:

1. The job goes into the job queue.
2. The dispatcher starts the job in turn, invoking the `mantas` process and passing the job ID and the thread count to the `mantas` process.
3. The `mantas` process creates the run entries in the metadata tables. Each job consists of one or more runs.
4. The `mantas` process handles the job runs.

After a job runs successfully, you can no longer copy, edit, or delete the job. The `start_mantas.sh` script waits for all jobs in the template group to complete.

Starting Jobs without the Dispatcher

Clients who use multiple services to run jobs for one database must run the jobs without dispatcher processes. If the client does use dispatchers on each machine, each dispatcher may run each job, which causes duplicate detection results.

To run a job template without a dispatcher, add the parameter `-nd` to the command line after the template ID. For example:

```
start_mantas.sh 100 -nd
```

This causes the `start_mantas.sh` script to execute all jobs in the template, rather than depending on the dispatcher to run them. The jobs in the template group run in parallel.

The dispatcher can ensure that it is only running a set number of max jobs at any given time (so if the max is set to 10 and a template has 20 jobs associated to it, only 10 run simultaneously). When running without the dispatcher, you must ensure that the number of jobs running do not overload the system. In the event a job run dies unexpectedly (that is, not through a caught exception but rather a fatal signal), you must manually verify whether any jobs are in the RUN state but do not have a `mantas` process still running, which would mean that the job threw a signal. You must update the status code to ERR to restart the job.

To start a new job without the **`dispatcher`**, follow these steps:

1. Create the new job and job description through the Oracle Financial Services Job Editor.

   The application automatically assigns a unique ID to the job when it is created.

2. Associate the job to a Job Template Group using the `KDD_JOB_TEMPLATE` table.

3. Execute the `start_mantas.sh` script with the following parameters:

   ```
   start_mantas.sh <template ID> [-sd DD-MON-YYYY]
   [-ed DD-MON-YYYY] [-nd]
   ```

   where the optional job parameters `-sd` and `-ed` (start date and end date, respectively) are used to constrain the data that an algorithm job pulls back.

   For example, if these parameters are passed into an Alert Creator job, the Alert Creator considers only matches for a grouping that has a creation date within the range that the parameters specify.

After a job runs successfully, you can no longer copy, edit, or delete the job.

Restarting a Job

Restarting a job is necessary when one or both of the following occurs:

- The dispatcher generates errors and stops during `mantas` processing. When the dispatcher is running, the Currency Transaction Reporting administrator can restart a job (or jobs) by changing each job's status code from ERR to RES.
- A job generates errors and stops during `mantas` processing. If a job stops processing due to errors, correct the problems that caused the errors in the job run and restart the job.

If the dispatcher stops, all jobs stop. You must restart the dispatcher and restart all jobs, including the job that generated real errors.

To restart a job, follow these steps:

Note: If the dispatcher has stopped, restart it.

1. Type `restart_mantas.sh <template group ID>` at the system prompt.

2. Press **Enter**.

   When the dispatcher picks up a job from the job queue that has a code of RES, it automatically restarts the job (See section *Starting Jobs*, on page 13, for more information).

   Note: By default, the `restart_mantas.sh` script looks for jobs run on the current day. To restart a job that was run on a specific date, you must provide the optional date parameter (for example, `restart_mantas.sh <template group ID> <DD-MON-YYYY>`).

Restarting Jobs without the Dispatcher

Restarting a job without the dispatcher is necessary when a job generates errors and stops during `mantas` processing. If a job stops processing due to errors, correct the problems that caused the errors in the job run and restart the job.

To start a new job, execute the `restart_mantas.sh` script with the following parameters:

```
restart_mantas.sh <template ID> [-sd DD-MON-YYYY] [-ed DD-MON-YYYY] [-nd]
```

Stopping Jobs

It may be necessary to stop one or more job processes when dispatcher errors, job errors, or some other event make it impossible or impractical to continue processing. In addition to stopping the processes, administrative intervention may have to resolve the cause of the errors.

To stop a job, you must stop its associated `mantas` process. To obtain the process IDs of active jobs and `mantas` processes:

1. Type **`ps -efw | grep mantas`** and press **Enter** at the system prompt.

   The **`mantas`** processes that are running appear on the computer screen as shown in the following example:

   ```
   00000306 7800 1843   0 Jul 16   ttyiQ/iAQM 0:00
    /kdd_data1/kdd/server/bin/mantas -j 123
   ```

   The `mantas` process ID number appears in the first display line in the second column from the left (7800). The job ID number appears in the second display line in the last column (-j 123).
2. Find the job and `mantas` process ID that you want to stop.
3. Type **`kill <mantas process ID>`** at the system prompt and press **Enter**.

   This command stops the `mantas` process ID, which also stops its associated job.

Monitoring and Diagnosing Jobs

In addition to the `dispatch.log` file that records events for all jobs, the system creates a job log for each job. A job log records only the events that are applicable to that specific job. By default, a job log resides in the `$KDD_PRODUCT_HOME/logs` directory. You can configure the location of this log in the `<INSTALL_DIR>/behavior_detection/algorithms/MTS/mantas_cfg/install.cfg` file.

If you do not know the location of the log directory, check the `install.cfg` file. The `log.mantaslog.location` property indicates the log location. The default is `$KDD_PRODUCT_HOME/logs`, but this location is configurable.

When troubleshooting a job processing problem, first look at the file `dispatch.log` for the sequence of events that occurred before and after errors resulted from a job. Then, look at the job log to diagnose the cause of the errors. The job log provides detailed error information and clues that can help you determine why the job failed or generated errors.

The log file name for a job appears in the following format in the log directory:

```
job<job_id>-<date>-<time>.log
```

where `<job_id>` is the job ID and `<date>` and `<time>` represent the job's starting timestamp.

If the job errors occurred due to a problem at the system level, you may need to resolve it. If you believe that the job errors were generated due to incorrect setups, you should notify the System Administrator, who can correct the problem setups.

Note: The `dispatch.log` may contain a JVM core dump. This does not indicate the actual cause of an error; you must See the job log for the underlying error.

To monitor a specific job or to look at the job log history for diagnostic purposes, follow the steps:

1. Type **`tail -f <log>`** at the system prompt and press **Enter**, where `<log>` is the name of the job log file.

   The job log scrolls down the screen.

2. Press **Ctrl+C** to stop the display.
3. Type **`lpr`** `job<job_id>-<date>-<time>` at the system prompt and press **Enter** to print the job log.

Caution: This job log file may be a lengthy printout.

Clearing Out the System Logs

Periodically, you need to clear out the dispatch and job log files. Otherwise, the files become so large that they are difficult to use as diagnostic tools and their size can impact the performance of the system.

Note: Oracle recommends that the Oracle client establish a policy as to the frequency for clearing the logs and whether to archive them before clearing.

Caution: Before you shut down the dispatcher to clear the system logs, verify that no jobs are active.

Clearing the Dispatch Log

To clear the `dispatch.log` file, follow the steps:

1. Shut down the `dispatcher` by following the procedure for Stopping the dispatcher (See section *Stopping the Dispatcher*, on page 11, for more information).
2. Type `cd <$KDD_PRODUCT_HOME>/logs` at the system prompt, where `<$KDD_PRODUCT_HOME>` is your product server installation directory.
3. Type `rm dispatch.log` to clear the dispatcher log.
4. Type **`start_chkdisp.sh <sleep time>`** and press **Enter** to restart the dispatcher.

Clearing the Job Logs

To clear the job logs, follow these steps:

1. Stop the `dispatcher` by following the procedure for Stopping the dispatcher (See section *Stopping the Dispatcher*, on page 11, for more information).
2. Type `cd <directory>` at the system prompt, where `<directory>` is your log directory.

By default, a job log resides in the directory `$KDD_PRODUCT_HOME/logs`. You can configure the location of this log in the `<INSTALL_DIR>/behavior_detection/algorithms/MTS/mantas_cfg/install.cfg` file.

If you do not know the location of the log directory, check the `install.cfg` file. The `log.mantaslog.location` property indicates the log location; the default is `$KDD_PRODUCT_HOME/logs` but this location is configurable.

3. Do either of the following:

- Type `rm job<job_id>-<date>-<time>.log` at the log directory prompt to clear one job log, where `<job_id>-<date>-<time>` is the name of a specific job log.
- Type `rm job*` to clear all job logs.

4. Restart the `dispatcher`.

Recovering Jobs from a System Crash

If the system crashes, all active jobs (`status_cd = RUN`) fail. You can recover the jobs by running the script `recover_mantas.sh`. This script changes the status_cd to RES so that these jobs can restart and finish running. The `recover_mantas.sh` script has an optional parameter—the date on which the system ran the `start_mantas.sh` script. This parameter has a `DD-MON-YYYY` format. The default value is the current date. Running the `recover_mantas.sh` script with this parameter ensures the script recovers only the jobs started that day. The dispatcher must be running to pick up the restarted jobs. This results in either a successful completion (`status_cd = FIN`) or failure (`status_cd = ERR`).

You can restart jobs that ended in failure by running the `restart_mantas.sh` script. The `restart_mantas.sh <template group ID>` script changes the status_cd from ERR to RES for any jobs passed in the template group that have a status_cd of ERR for the `dispatcher` to pickup.

CHAPTER 3 Security Configuration

This chapter provides instructions for setting up and configuring the Security Management System (SMS) to support Behavior Detection user authentication and authorization. It also contains instructions for setting up user accounts in the Behavior Detection database to access the Currency Transaction Reporting. This chapter focuses on the following topics:

- About Oracle Financial Services User Authentication
- About User Setup
- About Configuring Access Control Metadata
- Mapping Users To Access Control Metadata
- About Configuring Transmitter (1A) Record for E-File
- Setting the User Defined Home Page
- Configuring Exempt Filer Bank
- Configuring Affiliated Bank

About Oracle Financial Services User Authentication

The primary way to access Oracle Financial Services Behavior Detection information is through a Web browser that accesses Currency Transaction Reporting. Behavior Detection offers a built-in Authentication System and Security Management System (SMS) on the Web Application server that authenticates users from a login Web page. (See *Understanding SMS* for more information).

Understanding SMS

The Oracle Financial Services Behavior Detection SMS Engine is primarily responsible for user creation, maintenance, authentication, and authorization.

As an administrator, you can perform the following tasks:

- Create users
- Manage users
- Create user groups
- Map users to user groups
- Assign roles to user groups
- Create functions
- Map functions to roles

Accessing Oracle Financial Services Behavior Detection

A user gains access to Oracle Financial Services Behavior Detection based on the following:

- Authentication of a unique user ID and password that enables access to Oracle Financial Service Behavior Detection.
- Set of policies that associate their functional role with access to specific system functions in Oracle Financial Services Behavior Detection.
- Access to one or more jurisdictions.
- Access to one or more business domains.

About User Setup

To set up a user and provide the user access to Oracle Financial Services Behavior Detection, perform the following steps:

1. Create a user: See the *Oracle Financial Services Analytical Applications Infrastructure User Manual, Release 8.0.2* for setting up a user.
2. Once the user is created, map the user to the group. This in turn maps the user to the role. With this the user will have access to the privileges as per the role.

See *User Group and User Roles* for more information.

Note: See *Oracle Financial Services Analytical Applications Infrastructure User Manual, Release 8.0.2* for further information.

User Group and User Roles

The Oracle Financial Services User Roles are predefined in the Oracle Financial Services Behavior Detection application. Sample values for User groups are included in the installer but can be modified by clients to meet their specific needs. The corresponding mappings between User Roles and sample User Groups are predefined but can also be modified by clients to either adjust the role to sample user group mapping or to map roles to newly defined user groups.

For creating a new user group and mapping it to en existing role, Seethe following mentioned sections of the *Oracle Financial Services Analytical Applications Infrastructure User Manual, Release 8.0.2:*

- Defining User Group Maintenance Details
- Adding New User Group Details
- Mapping Users to User Group
- Mapping User Group(s) to Domain(s)
- Mapping User Group(s) to Role(s)

Actions to Role mappings are done through Database tables. Sample action to role mappings are included in the application.

- Working with Currency Transaction Reporting Action Settings

Actions are primarily associated with a User Role, not an individual user.

Table 9 describes the predefined User Roles and corresponding User Groups present in Oracle Financial Services Currency Transaction Reporting.

Table 9. CTR Roles and User Groups

Role	Group Name	User group Code
CTR Admin	CTR Admin Group	CTRADMNGR
CTR Analyst	CTR Analyst Group	CTRANALYST
CTR eFile Analyst	CTR EFile Ananlyst Group	CTREFALGR
CTR QA Analyst	CTR QA Analyst Group	CTRQAANGR
Supervisor	Supervisor Group	CTRSUPV
CTR Viewer	CTR Viewer Group	CTRVIEWER
Exemption Analyst	Exemption Analyst Group	EXMANALYST
DOEP Analyst and Supervisor	Exemption Analyst Group	DOEPEFLGRP

Mapping a User to a Single User Group

If a user is to have only one role then that user can be mapped to a single User Group associated with that User Role. See *Oracle Financial Services Analytical Applications Infrastructure User Manual, Release 8.0.2* to know more about *User to User Group mapping*.

Mapping a User to Multiple User Groups within Currency Transaction Report

If a user needs to have more than one role within Behavior Detection (that is, within Currency Transaction Reporting), then the user needs to be mapped to the different User Groups associated with the corresponding role. When the user logs into Oracle Financial Services, user access permissions would be the union of access and permissions across all roles.

Mapping a User to Multiple User Groups across Currency Transaction Report and Other Applications

If a user needs to have different roles in Currency Transaction Reporting and roles for other platform supported applications, then that user has to be mapped to different user groups. When such a user logs in, the user is taken to the Behavior Detection Start page, rather than Behavior Detection Home page. For any other platform applications the user is mapped to, clicking each link opens the selected application in a new window.

Defining the User Access Properties and Relationships

The following types of data compose a user's security configuration:

- **Business Domain(s):** Property that enables an Oracle client to model client data along operational business lines and practices.
- **Jurisdiction(s):** Property that enables an Oracle client to model client data across such attributes as geographic location or type or category of a business entity.
- **Role(s):** Permissions or authorizations assigned to a user in the system (such as, Oracle administrator or Auditor).

Table 10 provides the relationships between the data points that Figure 3 illustrates.

Table 10. Relationships between Data Points

Data Point	Relationship
Role	Associated with 0..n Users
	Has no direct relationship with an Organization
User	Associated with 1..n Business Domains
	Associated with 1..n Jurisdictions
	Associated with 1..n Roles
	Associated with 1..n Scenario Groups
	Associated with 1..n Case Type/Subtypes
	Associated with 1..n Organizations (as members)
	Associated with one Organization (as `mantasLineOrgMember`)
Business Domains	Associated to 0..n users
	Business domain *key* must be in the `KDD_BUS_DMN` table
Jurisdiction	Associated to 0..n users
	Jurisdiction *key* must exist in the `KDD_JRSDCN` table

Obtaining Information Before Configuring Access Control

Before you perform access control activities (for example, adding a group, modifying user information, or deleting a user), contact your system administrator for the following information to add to the locations in Table 11.

Table 11. Access Control Items and Locations

Data Item	Location
User Name	`KDD_REVIEW_OWNER`
User ID	`KDD_REVIEW_OWNER`
Role	`CSSMS_ROLE_MAST`
Business Domain	`KDD_BUS_DMN`
Jurisdiction	`KDD_JRSDCN`
Email Address	`KDD_REVIEW_OWNER`

Note: Email ID is mandatory for users who need to take Email actions. The user ID must be configured with valid email IDs through the User Maintenance UI.

About Configuring Access Control Metadata

You must first provide the user with access privileges, so the user can perform activities throughout various functional areas in Behavior Detection. This enables the user to access at least one of each of the following:

- **Jurisdiction:** Scope of activity monitoring for example, Geographical Jurisdiction or Legal entity
- **Business Domain:** Operational line of business

- **Role:** Permissions or authorizations assigned to a user.

Clients can change or add new values for these data types like jurisdiction, business domain (with the exception of User Role) based on specific requirements. The following section explains how to add or modify these data types.

Creating Jurisdiction in the Database

Behavior Detection uses Jurisdictions to limit user access to data in the database. Records from the Oracle client that ingestion loads must be identified with a jurisdiction, users of the system must be associated with one or more jurisdictions. In Currency Transaction Reporting, users can view data only associated with jurisdictions to which they have access. You can use a jurisdiction to divide data in the database; for example:

- **Geographical:** Division of data based on geographical boundaries, such as countries.
- **Organizational:** Division of data based on different legal entities that compose the client's business.
- **Other:** Combination of geographic and organizational definitions. In addition, it is client driven and can be customized.

In most scenarios, a jurisdiction also implies a threshold that enables use of this data attribute to define separate threshold sets based on jurisdictions.

Creating Jurisdiction in the Database through Scripts

You can create jurisdiction in the database using the following steps:

1. Add the appropriate record to the `KDD_JRSDCN` database table, which Table 12 describes.

Table 12. `KDD_JRSDCN` Table Attributes

Column Name	Description
`JRSDCN_CD`	Code (one to four characters) that represents a jurisdiction (for example, N for North, or S for South).
`JRSDCN_NM`	Name of the jurisdiction (for example, North or South).
`JRSDCN_DSPLY_NM`	Display name of the jurisdiction (for example, North or South).
`JRSDCN_DESC_TX`	Description of the jurisdiction (for example, Northern US or Southern US).

2. Add records to the table by using a SQL script

```
INSERT INTO KDD_JRSDCN (JRSDCN_CD, JRSDCN_NM,
JRSDCN_DSPLY_NM,
```

Figure 3. Sample SQL Script for Loading KDD_JRSDCN

Note: The `KDD_JRSDCN` table is empty after system initialization and requires populating before the system can operate.

Creating Business Domain

Business domains are used for data access controls similar to jurisdiction but have a different objective. The business domain can be used to identify records of different business types (for example, Private Client vs. Retail customer), or to provide more granular restrictions to data such as employee data. The list of business domains in the system resides in the KDD_BUS_DMN table. Behavior Detection tags each data record provided through data ingestion to one or more business domains. Behavior Detection also associates users with one or more business domains in a similar fashion. If a user has access to any of the business domains that are on a business record, the user can view that record.

The business domain field for users and data records is a multi-value field. For example, you define two business domains:

- **a:** Private Client
- **b:** Retail Banking

A record for an account that is considered both has BUS_DMN_SET=ab. If a user can view business domain **a** or **b**, the user can view the record. You can use this concept to protect special classes of data, such as data about executives of the firm. For example, you can define a business domain as *e: Executives*.

You can set this business domain with the employee, account, and customer records that belong to executives. Thus, only specific users of the system have access to these records. If the executive's account is identified in the Private Client business domain as well, any user who can view Private Client data can view the executive's record. It is important not to apply too many domains to one record.

Creating Business Domain in the Database through scripts

To create a business domain, follow these steps:

1. Add the appropriate user record to the KDD_BUS_DMN database table, which Table 13 describes.

Table 13. KDD_BUS_DMN Table Attributes

Column Name	Description
BUS_DMN_CD	Single-character code that represents a business domain (for example, a, b, or c).
BUS_DMN_DESC_TX	Description of the business domain (for example, Institutional Broker Dealer or Retail Banking).
BUS_DMN_DSPLY_NM	Display name of the business domain (for example, INST or RET).
MANTAS_DMN_FL	Flag that indicates whether Behavior Detection specified the business domain (Y). If an Currency Transaction Reporting client specified the business domain, you should set the flag to N.

The KDD_BUS_DMN table already contains predefined business domains for the Currency Transaction Reporting client.

2. Add more records to the table by using a SQL script similar to the sample script in Figure 4.

```
INSERT INTO KDD_BUS_DMN (BUS_DMN_CD, BUS_DMN_DESC_TX,
BUS_DMN_DSPLY_NM, MANTAS_DMN_FL) VALUES ('a', 'Compliance
Employees', 'COMP', 'N');

INSERT INTO KDD_BUS_DMN (BUS_DMN_CD, BUS_DMN_DESC_TX,
BUS_DMN_DSPLY_NM, MANTAS_DMN_FL) VALUES ('b', 'Executives',
'EXEC', 'N');
```

Figure 4. Loading the KDD_BUS_DMN Table

3. Update the KDD_CENTRICITY table to reflect access to all focuses within the business domain with the following command:

```
update KDD_CENTRICITY set bus_dmn_st = 'a'
where KDD_CENTRICITY. CNTRY_TYPE_CD = 'SC'
```

Mapping Users To Access Control Metadata

An Administrator can map each user to Access Control Metadata and Security attributes which will control the user's access permissions. In order to provide this mapping to each user an entry is needed to be made in KDD_REVIEW_OWNER table of mantas schema usingthe following query

Table 14. KDD_REVIEW_OWNER table Attributes

Column Name	Description
OWNER_SEQ_ID	Unique identifier of the User.
ACTV_FL	Indicator of whether this user is currently active.
OWNER_DSPLY_NM	The user displayname
OWNER_ID	Logon name of this user
OWNER_TYPE_CD	Type of user
CURR_VALID_LOGON_TS	Date and time that this user logged on for the most recent session.
EMAIL_ADDR_TX	Email address of the user
LAST_FAILED_LOGON_TS	Date and time of the last unsuccessful login attempt for this user
OWN_ALERT_FL	Indicator of whether this owner can own an alert (not required for Currency Transaction Reporting)
OWN_CASE_FL	Indicator of whether this owner can own a case (not required for Currency Transaction Reporting)
PREV_VALID_LOGON_TS	Date and time that this user logged on prior to the current session.
RPTG_GROUP_CD	Name of the organization to which this user belongs/reports. (not required for Currency Transaction Reporting)
BUS_DMN_ST	Set of business domains to which this user has access.

```
INSERT INTO KDD_REVIEW_OWNER  (OWNER_SEQ_ID,
   OWNER_ID,
   OWNER_TYPE_CD,
   RPTG_GROUP_CD,
   ACTV_FL,
   BUS_DMN_ST,
          EMAIL_ADDR_TX,
          OWNER_DSPLY_NM,
          OWN_ALERT_FL,
          LAST_FAILED_LOGON_TS,
          CURR_VALID_LOGON_TS,
          PREV_VALID_LOGON_TS,
          OWN_CASE_FL)
          SELECT  F_GET_NEXT_VAL('OWNER_SEQ_ID_SEQ'),
                  A.V_USR_ID,
                  'USER',
                  NULL,
                  CASE WHEN A.F_USR_DELETE = 'Y' THEN
                       'N'
                     WHEN TO_DATE(SUBSTR(A.D_USR_EXPIRY_DTE, 0, 10),'MM/DD/YYYY')
< SYSDATE THEN
                       'N'
                     WHEN TO_DATE(SUBSTR(A.D_USR_EXPIRY_DTE, 0, 10),'MM/DD/YYYY')
> SYSDATE THEN
                       'Y'
                      ELSE
                       A.F_USR_ENABLED
                  END ACTV_FL,
                  '<BUSINESS DOMAIN>',
                  A.V_EMAIL,
                  A.V_USR_NAME,
                  'N',
                  NULL,
                  NULL,
                  NULL,
                  'N'
    FROM CSSMS_USR_PROFILE A
   WHERE A.V_USR_ID = <Any User ID listed in the Owner_ID in the kdd_review_owner
table>
      COMMIT;
```

For example

An entry has to be made for user with Id and business domains as x, y, z so the query would be

```
INSERT INTO KDD_REVIEW_OWNER  (OWNER_SEQ_ID,
   OWNER_ID,
   OWNER_TYPE_CD,
   RPTG_GROUP_CD,
   ACTV_FL,
   BUS_DMN_ST,
          EMAIL_ADDR_TX,
          OWNER_DSPLY_NM,
          OWN_ALERT_FL,
          LAST_FAILED_LOGON_TS,
          CURR_VALID_LOGON_TS,
          PREV_VALID_LOGON_TS,
          OWN_CASE_FL)
          SELECT  F_GET_NEXT_VAL('OWNER_SEQ_ID_SEQ'),
                  A.V_USR_ID,
                  'USER',
                  NULL,
                  CASE WHEN A.F_USR_DELETE = 'Y' THEN
```

```
                        'N'
                      WHEN TO_DATE(SUBSTR(A.D_USR_EXPIRY_DTE, 0, 10),'MM/DD/YYYY')
< SYSDATE THEN
                        'N'
                      WHEN TO_DATE(SUBSTR(A.D_USR_EXPIRY_DTE, 0, 10),'MM/DD/YYYY')
> SYSDATE THEN
                        'Y'
                       ELSE
                        A.F_USR_ENABLED
                  END ACTV_FL,
                  'xyz',
                  A.V_EMAIL,
                  A.V_USR_NAME,
                  'N',
                  NULL,
                  NULL,
                  NULL,
                  'N'
    FROM CSSMS_USR_PROFILE A
   WHERE A.V_USR_ID = 'USER1'
     COMMIT;
```

Components of Security Attribute

After the data is made available in the `KDD_REVIEW_OWNER` table then the user needs to be mapped to the following parameters

- Jurisdiction
- Business Domain

Jurisdiction

Mapping of one or more jurisdictions to a user or organization, gives the privilege of accessing Currency Transaction Reporting records that belong to the mapped jurisdiction. This is done by executing the following query in the atomic schema:

```
INSERT INTO KDD_REVIEW_OWNER_JRSDCN(OWNER_SEQ_ID,JRSDCN_CD)
VALUES(10153,'AMEA');
COMMIT;
```

Business Domain

Mapping of one or more business domains to a user or organization gives the user privileges to access CTR records that belong to the mapped business domains. This is done when an entry is made in the KDD_REVIEW_OWNER table (See *Mapping Users To Access Control Metadata,* on page 25,). In order to modify the business domain of any user execute the following query in the atomic schema

```
UPDATE KDD_REVIEW_OWNER
SET BUS_DMN_ST ='abc'
```

```
WHERE OWNER_ID = <USER_ID>
```

where **a,b, c** are valid business domains and **USER_ID** is the ID (as in `KDD_REVIEW_OWNER` table `OWNER_ID` column) of the user to which domain mapping is to be done.

About Configuring Transmitter (1A) Record for E-File

The first record on each E-File must be the Transmitter (1A) Record and to provide this the corresponding table is `KDD_TRANSMITTER`. This table contains information identifying the batch file transmitter (person or organization handling the data accumulation and formatting). There will be only one transmitter record in the table. All data elements for this record are required.

Table 15. KDD_TRANSMITTER table Attributes

Column Name	Description
TRNSMTR_NM	Client Company Name (Static)
TRNSMTR_ADDR_STRT_TX	Client Company Address (Static)
TRNSMTR_ADDR_CITY_NM	Client Company City (Static)
TRNSMTR_ADDR_STATE_CD	Client Company State (Static)
TRNSMTR_ADDR_POSTL_CD	Client Company ZIP Code (Static)
TRNSMTR_ADDR_CNTRY_CD	Client Company Country (Static)
TRNSMTR_PHON_NB	Client Company Telephone Number(Static)
TRNSMTR_CNT_NM	Client Company Contact Name for CTR (Static)
TRNSMTR_TAX_ID	Client Company TIN (Static)
TRNSMTR_CNTRL_CD	8-character Transmitter Control Code
CNT_OFFICE_NM	Name of the office to contact for information concerning the BSA CTR
PHON_NB	Contact office phone number
PHON_EXT_NB	Contact office phone extension

In order to insert a record in the `KDD_TRANSMITTER` table, execute the following query in the Atomic schema

```
INSERT INTO KDD_TRANSMITTER(TRNSMTR_NM,
                            TRNSMTR_ADDR_STRT_TX,
                            TRNSMTR_ADDR_CITY_NM,
                            TRNSMTR_ADDR_STATE_CD,
                            TRNSMTR_ADDR_POSTL_CD,
                            TRNSMTR_ADDR_CNTRY_CD,
                            TRNSMTR_PHON_NB,
                            TRNSMTR_CNT_NM,
                            TRNSMTR_TAX_ID,
                            TRNSMTR_CNTRL_CD,
                            CNT_OFFICE_NM,
                            PHON_NB,
                            PHON_EXT_NB)
VALUES ('<Name>',
       '<Address>',
       '<City>',
       '<State>',
       <Postal Code>,
       '<Country>',
       <Phone Number>,
       '<Contact Name>',
       <Company TIN>,
       <Transmitter Code>,
       '<Contact Office Name>',
       <Contact Office Phone Number>,
       <Contact Office Extension>);
COMMIT;


For example:-
INSERT INTO KDD_TRANSMITTER(TRNSMTR_NM,
                            TRNSMTR_ADDR_STRT_TX,
                            TRNSMTR_ADDR_CITY_NM,
                            TRNSMTR_ADDR_STATE_CD,
                            TRNSMTR_ADDR_POSTL_CD,
                            TRNSMTR_ADDR_CNTRY_CD,
                            TRNSMTR_PHON_NB,
                            TRNSMTR_CNT_NM,
                            TRNSMTR_TAX_ID,
                            TRNSMTR_CNTRL_CD,
                            CNT_OFFICE_NM,
                            PHON_NB,
                            PHON_EXT_NB)
VALUES ('JPMC',
       '12bdfg',
       'bangalore',
       'kar',
       560038,
       'IN',
       12344567,
       'HSDN',
       12345678,
       45678910,
       'CONTACT OFC',
       123678,
       0532);
COMMIT;
```

Some financial institutions for which parent financial institution (2A record)identification is not available then we need to have default parent institution identification in `KDD_ORG` table for all such CTR records.

Table 16. KDD_ORG table Attributes

Column Name	Description
`CTR_ID`	Identifier for a specific Currency Transaction Reporting
`ORG_INTRL_ID`	Identifier for a specific organization that is unique across the enterprise.

Table 16. KDD_ORG table Attributes

DATA_DUMP_DT	Business date for which the data record is provided to Oracle Financial Services.
ORG_NM	Name of this organization.
ORG_SEQ_ID	Oracle Financial Services-specific identifier for this organization that is unique
ORG_TYPE_CD	Identifier of the Oracle client-specified type of this organization
ALT_ORG_INTRL_ID	Alternative identifier for this organization that is unique across the enterprise.
COST_CTR_ID	Cost center to which the Oracle client assigns this organization.
SRC_SYS_CD	Source system from which this data content was extracted.
ORG_CNTRY_CD	Country where the organization is located or headquartered.
CSTM_1_DT	Date field that is available for use at the Oracle client’s discretion.
CSTM_2_DT	Date field that is available for use at the Oracle client’s discretion.
CSTM_3_DT	Date field that is available for use at the Oracle client’s discretion.
CSTM_1_RL	Number field that is available for use at the Oracle client’s discretion.
CSTM_2_RL	Number field that is available for use at the Oracle client’s discretion.
CSTM_3_RL	Number field that is available for use at the Oracle client’s discretion.
CSTM_1_TX	Text field that is available for use at the Oracle client’s discretion.
CSTM_2_TX	Text field that is available for use at the Oracle client’s discretion.
CSTM_3_TX	Text field that is available for use at the Oracle client’s discretion.
CSTM_4_TX	Text field that is available for use at the Oracle client’s discretion.
CSTM_5_TX	Text field that is available for use at the Oracle client’s discretion.
PRCSNG_BATCH_NM	Ingestion batch in which Oracle Financial Services processed this data record.
JRSDCN_CD	Jurisdiction associated with this organization.
BUS_DMN_LIST_TX	Organization's business domain(s); for example, institutional or retail brokerage. Oracle Financial Services uses this field to control access to data across distinct business operations.
ORG_GRP_ID	Financial identification number for the Organization where the organization represents a financial institution.
ORG_DIVSN_ID	Financial identification number for the division of the Organization where the organization represents a financial institution.
ORG_PRMRY_FED_REG_CD	The parent financial institution primary federal regulator code for the federal regulator or BSA examiner with primary responsibility for enforcing the institution's Bank Secrecy Act compliance.
ORG_LEGAL_NM	The full legal name of the parent financial institution
ORG_ALT_NM	The financial institution alternate name
ORG_EIN_NB	The parent financial institution’s EIN. If the financial institution does not have an EIN, enter the SSN of the institution’s principal owner. Do not enter hyphens, slashes, alpha characters, or invalid entries such as all nines, all zeros, or “123456789”
ORG_ADDR_STRT_TX	The address of the parent financial institution headquarters
ORG_ADDR_CITY_NM	Enter the city of the parent financial institution headquarters.

Table 16. KDD_ORG table Attributes

ORG_ADDR_STATE_CD	Enter the code for the parent financial institution headquarters stat
ORG_ADDR_POSTL_CD	The parent financial institution ZIP Code. Do not include punctuation or formatting such as hyphens, periods, and spaces within the entry. A 9-digit entry cannot end with four zeros
ORG_FINCL_ID	Financial identification number for the Organization where the organization represents a financial institution.
ORG_FINCL_ID_TYPE_CD	Type of identification for the Organization where the organization represents a financial institution.
FINCL_ID_TYPE_OTHER_DESC_TX	Description of Organization Type if it is equal to Z (Other).
PARENT_ORG_FL	Indicator to specify that this organization is a parent, if the value is 'Y'.
LAST_UPDATE_TS	This specifies the time stamp when the record was last update

Where Parent FI is not present, a default parent FI value for all such CTRs can be provided using the following Insert script.

```
INSERT INTO KDD_ORG (CTR_ID,
                            ORG_INTRL_ID,
                            DATA_DUMP_DT,
                            ORG_NM,
                            ORG_SEQ_ID,
                            ORG_TYPE_CD,
                            ALT_ORG_INTRL_ID,
                            COST_CTR_ID,
                            SRC_SYS_CD,
                            ORG_CNTRY_CD,
                            CSTM_1_DT,
                            CSTM_2_DT,
                            CSTM_3_DT,
                            CSTM_1_RL,
                            CSTM_2_RL,
                            CSTM_3_RL,
                            CSTM_1_TX,
                            CSTM_2_TX,
                            CSTM_3_TX,
                            CSTM_4_TX,
                            CSTM_5_TX,
                            PRCSNG_BATCH_NM,
                            JRSDCN_CD,
                            BUS_DMN_LIST_TX,
                            ORG_GRP_ID,
                            ORG_DIVSN_ID,
                            ORG_PRMRY_FED_REG_CD,
                            ORG_LEGAL_NM,
                            ORG_ALT_NM,
                            ORG_EIN_NB,
                            ORG_ADDR_STRT_TX,
                            ORG_ADDR_CITY_NM,
                            ORG_ADDR_STATE_CD,
                            ORG_ADDR_POSTL_CD,
                            ORG_FINCL_ID,
                              ORG_FINCL_ID_TYPE_CD,
                              FINCL_ID_TYPE_OTHER_DESC_TX,
                              PARENT_ORG_FL,
                              LAST_UPDATE_TS)
 VALUES(-1,
       < organization ID >,
       NULL,
       <organization Name>,
       NULL,
            NULL,
            NULL,
       NULL,
       NULL,
       NULL,
       NULL,
       NULL,
       NULL,
       NULL,
       NULL,
       NULL,
       NULL,
       NULL,
       NULL,
       NULL,
       NULL,
       NULL,
       NULL,
       NULL,
       NULL,
       NULL,
       < Institution Primary Federal Regulator ID>,
       NULL,
       NULL,
       <Institution EIN>,
       < Institution Address >,
       < Institution City >,
       < Institution State >,
       <Institution ZIP Code>,
       NULL,
       NULL,
       NULL,
       'Y',
       NULL
)
```

Setting the User Defined Home Page

Follow these steps to set the user defined home page:

1. Click **Home** from the LHS menu when you log into OFSAAI for the first time.
2. An option to select the default screen for the user appears.

3. Select **CTR** from the drop-down list, and click **Save.**
4. The CTR home page is set as the landing page for the user.

Configuring Exempt Filer Bank

The information in this table is used to populate information for the Exempt Filer Bank on the CTR. It is expected to have a single row to represent the filing bank.

Table 17. KDD_FILER table Attributes

Column Name	Description
`FILER_BANK_NM`	Name of the filing bank.
`FILER_BANK_EIN_NB`	EIN ID of the filing bank.
`FILER_BANK_RSSD_NB`	RSSD ID of the filing bank.
`FILER_BANK_ADDR`	Street address of the filing bank.
`FILER_BANK_ADDR_CITY`	City of the filing bank.
`FILER_BANK_ADDR_STATE_CD`	State code of the filing bank. If a US state/territory, then it must adhere to the two-letter codes used by the United States Postal Service when the country is equal to US or a U.S. Territory.
`FILER_BANK_ADDR_POSTL_CD`	The zip code or postal code for the filing bank. Do not include punctuation or formatting such as hyphens, periods, and spaces within the entry.
`PRIMARY_FEDERAL_REGULATOR`	Enter the appropriate code to record the filing institution's primary federal regulator. The value provided must adhere to the following requirements: 1 FRB (Federal Reserve Board) 2 FDIC (Federal Deposit Insurance Corporation) 7 IRS (Internal Revenue Service) 3 NCUA (National Credit Union Administration) 4 OCC (Office of the Comptroller of the Currency) Record the code (1,2,7,3,4) only.
`AFFILIATED_BANKS_INDICATOR`	Indicates that a parent bank or financial holding company (or one of its bank subsidiaries) is making the designation of exempt person on behalf of some or all bank subsidiaries of the holding company.
`FILER`	Filer requesting the exemption.
`USER_FIELD`	User who approved the exemption.

For example: Insert into `KDD_FILER`

```
('INSTITUTION','111122220',null,'ABCLANE',
'CITY','STATE','53002', null,
null, 'FILING INSTITUTION, null)
```

Configuring Affiliated Bank

The information in this table is used to populate data related to any Exempt Affiliated Banks that may be associated with the Exempt Filer Bank. The values in this table are used to allow a user to select the relevant affiliated bank from the Exemption record in the UI and have the associated information automatically populate on the Exemption record and DOEP.

Table 18. KDD_AFILIATED_BANK Table Attributes

Column Name	Description
`AFILIATED_BANK_NM`	Name of the affiliated bank.
`FILER_EIN_NB`	EIN ID of the filer.
`AFILIATED_BANK_EIN_NB`	EIN ID of the affiliated bank.
`AFILIATED_BANK_RSSD_NB`	RSSD ID of the affiliated bank.
`AFILIATED_BANK_ADDR`	Street address of the affiliated bank.
`AFILIATED_BANK_ADDR_CITY`	City of the affiliated bank.
`AFILIATED_BANK_ADDR_STATE_CD`	State code of the affiliated bank. If a US state/territory, then it must adhere to the two-letter codes used by the United States Postal Service when the country is equal to US or a U.S. Territory.
`AFILIATED_BANK_ADDR_POSTL_CD`	The zip code or postal code for the affiliated bank. Do not include punctuation or formatting such as hyphens, periods, and spaces within the entry.
`PRIMARY_FEDERAL_REGULATOR`	Enter the appropriate code to record the filing institution's primary federal regulator. The value provided must adhere to the following requirements: 1 FRB (Federal Reserve Board) 2 FDIC (Federal Deposit Insurance Corporation) 7 IRS (Internal Revenue Service) 3 NCUA (National Credit Union Administration) 4 OCC (Office of the Comptroller of the Currency) Record the code (1,2,7,3,4) only.
`AFFILIATED_BANKS_INDICATOR`	Indicates that a parent bank or financial holding company (or one of its bank subsidiaries) is making the designation of exempt person on behalf of some or all bank subsidiaries of the holding company.
`FILER`	Filer requesting the exemption.
`USER_FIELD`	User who approved the exemption.

For example: Insert into `KDD_AFILIATED_BANK`

```
('BANK', '111122220', '222', null,'ABC STREET',
'CITY', 'STATE','55555', null, null, null))
```

CHAPTER 4

Data Ingestion

This chapter explains the concept of data management and provides step-by-step instructions on how to load and process data for the CTR application. Specifically, this chapter focuses on the following topics:

- About Data Management
- Data Loading and Processing Flow Overview
- Managing Data Loading
- BDF.xml File Parameters
- BDF Directory Structure
- Alternatives to Standard Data Management Practices

About Data Management

Data Management consists of two main activities:

- **Data Loading**: Data is loaded from the Common Staging Area (CSA) and Flat File Interface into the Financial Services Data Model (FSDM) using various approaches such as Analytical Applications Infrastructure Table-to-table (AAI T2T), Analytical Applications Infrastructure Hive-to-table (AAI H2T), Behavior Detection Framework (BDF), and Run DP (Data Processing)/Run DL (Data Loading).
- **Data Processing**: The loaded data is processed (data aggregation/data derivation) using BDF datamaps in FSDM. The processing refers to the wide range of activities to include data enrichment and data transformation.

In addition to the file adapter interface to accept the data, FCCM also supports the OFSAA Common Staging Area (CSA).

Data Loading and Processing Flow Overview

The following figure provides an overview of the data loading and processing flow:

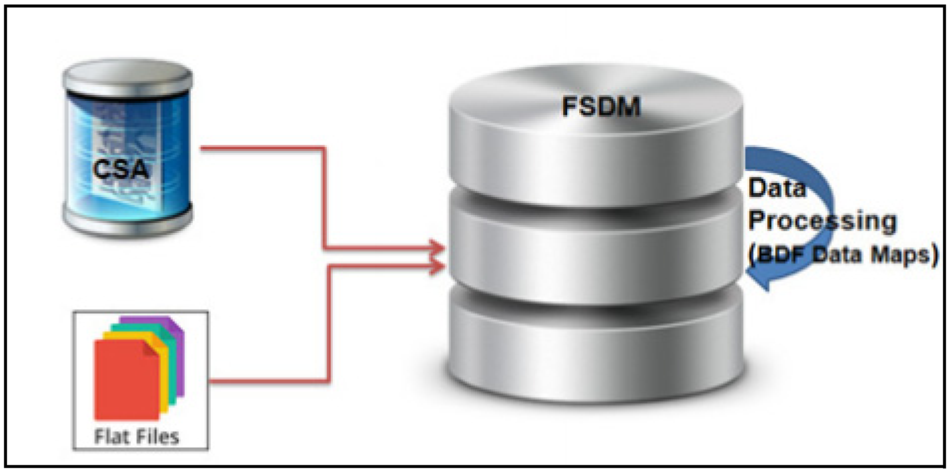


Figure 5. Data Loading and Processing Flow Overview

In BD applications, data is loaded through the CSA, hive, or flat files. It is then processed using BDF datamaps in the FSDM.

Common Staging Area (CSA)

The CSA provides a single repository for data storage for multiple functional areas and applications having the Common Staging Area Model and Reporting Data Model. The Common Staging Area Model provides a simplified, unified data sourcing area for inputs required by FCCM using Oracle Financial Services Behavior Detection Framework (OFSBDF).

Flat Files

The flat files contain data provided by the client. This data is loaded into the Financial Services Data Model (FSDM). The FSDM is a database which consists of well organized business data for analysis. It determines the structured data which stores persistent information in a relational database and is specified in a data modeling language.

BDF Datamaps

The BDF datamaps load Business, Market and Reference data required for alert processing. It does the data derivation and aggregation after Data Ingestion loads the base tables.

Managing Data Loading

Data loading is the process of loading raw or business data from the CSA or the flat file interface to the FSDM.

The following approaches are used to load the data:

- Loading Data to FSDM through CSA
- Loading Data to FSDM through Flat File Interface
- Generating Change Logs with BDF

Loading Data to FSDM through CSA

This section covers the following topics:

- Overview
- Loading T2T using the AAI Framework
- Using Behavior Detection Framework

Overview

In the CSA approach, you can load data using AAI Framework (T2T/H2T) and BDF. The following figure provides an overview of the data loading flow using CSA:

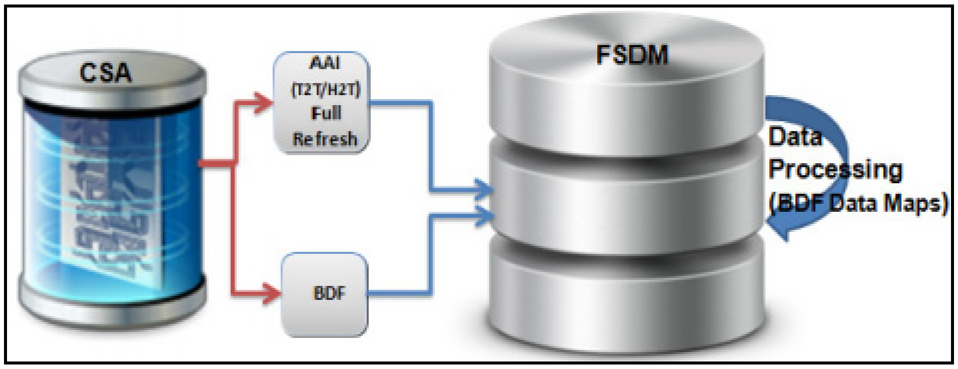


Figure 6. Data Management Flow Using CSA

Loading T2T using the AAI Framework

Table-to-Table (T2T) is used in the AAI Framework for data loading. The source for T2T data is the Oracle RDBMS.

About AAI T2T Data Loading

AAI (Analytical Applications Infrastructure) is a complete end-to-end Business Intelligence solution. It is a single interface that lets you access your company's operational data and use it to track and respond to business trends. It also facilitates the analysis of processed data.

The AAI framework is the process of retrieving structured data from data sources for further data processing, storage, or migration. The intermediate extraction process is followed by data transformation and metadata addition before exporting it to the Business Data Model. For more information, see *Chapter 2*, *Section - Data Mapping, Oracle Financial Services Analytical Applications Infrastructure User Guide*.

This section covers the following topics:

- Process Flow for AAI T2T
- Starting a Batch
- Verifying the Data Quality Using DQ
- Executing Data Transformation using DT
- Moving Data through T2T
- Executing Behavior Detection Jobs
- Ending the Batch

Process Flow for AAI T2T

The following figure shows the process flow for AAI T2T:

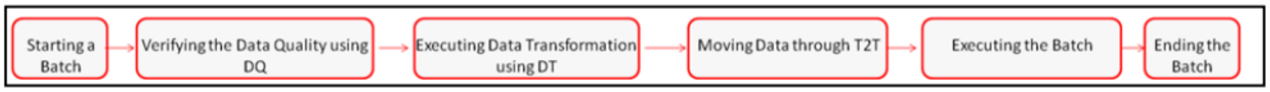


Figure 7. Process Flow for AAI T2T

Starting a Batch

Note: Ensure that the staging data has the same batch date records.

To start the batch, run the `start_mantas_batch.sh` and `set_mantas_date.sh` scripts. For more information, see *Managing Batch Control Utility*.

Verifying the Data Quality Using DQ

Data Quality (DQ) is a check that is done at every level based on the FSDM table. When data is moved from CSA to FSDM, a check is done on CSA. This check is done in order to move only useful data into the FSDM table. For example, a column in FSDM should not be blank if it is mandatory. These checks are also called rules.

The following data quality checks are done:

- **Length Validation Check**: If the length of data in source column is more than the length of target column, then an error message is generated. For example, if the `ACCT_INTRL_ID` column, which has a column length of 50 characters, needs to be populated from the source table column `V_ACCOUNT_NUMBER`, which has a few data with length more than 50 characters. An error message is raised.
- **Domain Check**: If any data does not qualify for the domain values, then an error message is generated. For example, if the valid value that column `ADDR_USAGE_CD` accepts is one of M|B|L|A|O|P|D|H|X|V, but the source column `V_ADDRESS_PURPOSE_TYPE_IND` has additional values such as E or C. An error message is raised.
- **Mandatory Check**: If a column which must have a value for the record to be valid has a null value, then an error message is generated. For example, if the column `ADDR_STRT_LINE1_TX` needs to have a value for the record to be valid and is mapped to the source table column `V_ADDRESS_LINE1`. If the column ADDR_STRT_LINE1_TX has a null value, an error message is raised.
- **Threshold Test**: If a target table column must have a value that is greater than 0 but has a value of 0, then an error message is generated. For example, if the target table column `LDGR_AM` must have a value that is greater than 0 but the source table column `N_LEDGER_BAL` has a value as 0 or null, an error message is raised.

Note: In addition to the above data checks, another data check is done for duplicate data during data loading through AAI T2T.

Executing Data Transformation using DT

The Data Transformation (DT) functionality allows you to delete the existing data in the AAI. For more information, see *Ways of Data Loading*.

Moving Data through T2T

Data is exported or moved from the CSA to the FSDM using AAI T2T. For more information on moving data through T2T, see *Chapter 2*, *Section - Data Mapping* of the *Oracle Financial Services Analytical Applications Infrastructure User Guide*.

Executing Behavior Detection Jobs

After the data quality check, data transformation, and data movement through T2T is completed, execute the following jobs:

- BDF Transformation jobs. For more information, see *BDF Derived Datamap Types*.
- Scenario jobs. For more information, see *Managing Scenario Migration Utility*.
- Scenario post-processing jobs. For more information, see *Post-Processing Tasks*.

Ending the Batch

To end the batch, run the `end_mantas_batch.sh` script. For more information, see *Managing Batch Control Utility*.

Using Behavior Detection Framework

The Behavior Detection Framework (BDF) datamap takes the data from the CSA, enhances it, and then loads it into a target database table (FSDM). The Data Interface Specification (DIS) datamaps are used to load client-provided data, either through DIS files as specified in the DIS or through CSA tables.

Note: All the DIS datamaps in the Behavior Detection Flat File Interface for which staging representation is marked as *Yes* are applicable for CSA loading. For more information, see *Behavior Detection Flat File Interface*.

To load data in the FSDM using BDF, follow these steps:

1. Configure the `DIS.source` parameter to FSDW. For more information on configuring other parameters, see *Behavior Detection Flat File Interface*.
2. Execute the Account datamap which loads data into the Account (ACCT) table using the following sample script:

   ```
   <OFSBDF Installed Directory>/bdf/scripts/execute.sh Account
   ```

   The above step can be repeated for all datamaps for which staging representation is marked as *Yes*.

Note: If there are any errors or rejections in loading data, refer to the `<OFSBDF Installed Directory>/bdf/logs` path to know about the errors in the log file.

Loading Data to FSDM through Flat File Interface

The loading process receives, transforms, and loads Market, Business, and Reference data that alert detection and assessment investigation processing requires. After loading the base tables, the Oracle client's job scheduling system invokes BDF datamaps to derive and aggregate data.

This section covers the following topics:

- Overview
- Using Behavior Detection Framework (Known as BDF datamaps)
- Ways of Data Loading (Known as runDP- runDL)

Overview

The following figure provides an overview of the data management flow using Flat File Interface:

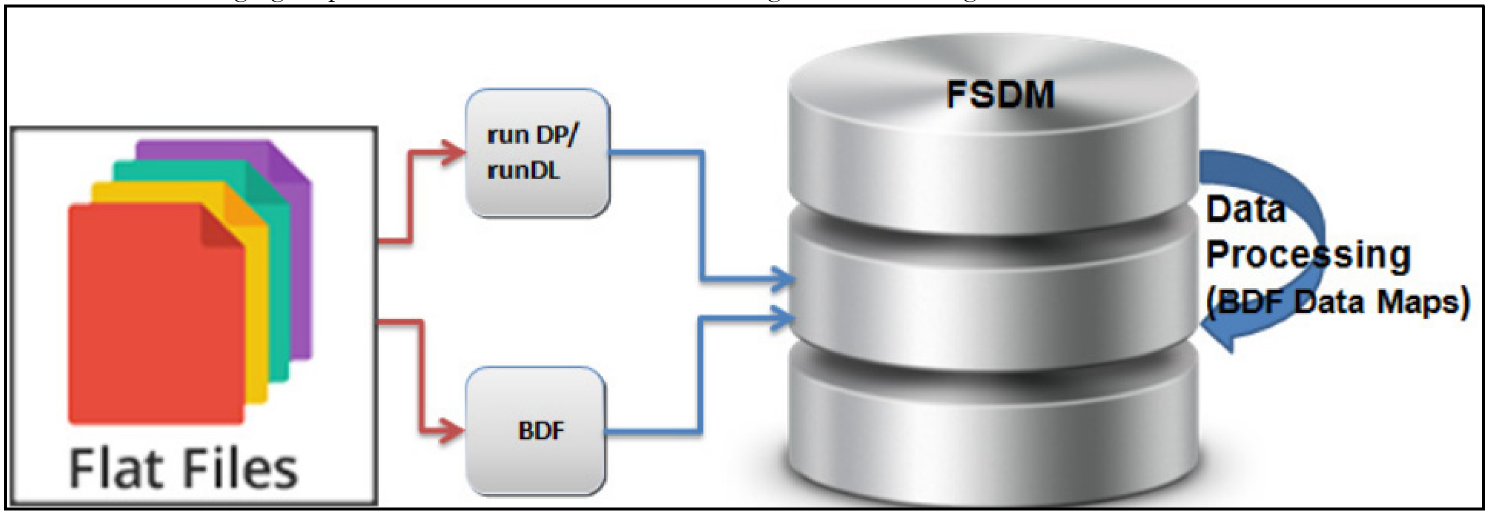


Figure 8. Data Loading Flow Using Flat File Interface

Note: All DIS datamaps in the Behavior Detection Flat File Interface for which staging representation is marked as *Yes* are applicable for Flat File loading. For more information, see *Behavior Detection Flat File Interface*.

Using Behavior Detection Framework

The Behavior Detection Framework (BDF) datamap takes the data from the flat files, enhances it, and then loads it into a target database table (FSDM).

To load data in the FSDM using Flat Files, follow these steps:

1. Place the `ASCII.dat` flat files in the `<OFSAAI Installed Directory>/bdf/inbox` directory.
2. Configure the `DIS.source` parameter to FILE. For more information on configuring other parameters, see *Appendix D, "Managing Data"*.
   - Configure the `DIS.Source` parameter to `FILE-EXT` for loading flat files through the external table. In order to load the flat files using the external table, the `ext_tab_dir_path` variable must also be set to the inbox directory and the database UNIX account must have read and write privileges to it.
3. Execute the Account datamap which loads into the Account (ACCT) table:

```
<OFSBDF Installed Directory>/bdf/scripts/execute.sh Account
```

Note: If there are any errors in loading, refer to the `<OFSBDF Installed Directory>/bdf/logs` path.

Ways of Data Loading

This section covers the following topics:

- Full Refresh Data Loading
- Incremental (Delta) Data Loading

Note: The following ways of data loading is applicable only for DIS files defined with load operation as Overwrite.

Full Refresh Data Loading

For full refresh data loading, first data is truncated and then new data is inserted. For example, suppose five records are loaded on Day 1. If new data is required on Day 2 based on the business keys defined on the DIS files, a full refresh data load can be done.

To do a full refresh data load, set `load.fullrefresh` to true in the `<OFSAAI Installed Directory>/bdf/config/BDF.xml` path. .

The time taken to do a full refresh data load is less than for an incremental load, although complete data must be provided every time.

Incremental (Delta) Data Loading

For incremental data loading, the following can be done:

- Data can be merged
- Existing data can be updated
- New data can be inserted

For example, suppose five records are loaded on Day 1. If four new records need to be inserted and one existing record needs to be updated based on the business keys defined on the DIS files, an incremental data load can be done.

To do an incremental data load, set `load.fullrefresh` to false in the `<OFSAAI Installed Directory>/bdf/config/BDF.xml` path.

Note: The time taken to do an incremental data load is more than for a full refresh data load, although there is no need to give complete data every time. Only updated or new data is required.

Note: As an Oracle client loading data through CSA, the system groups various source systems into one processing batch, so that it can call upon a specific batch and load data from specific source systems within that batch. This allows the handling of different batch loads from different countries running on the same staging instance. The association of the source systems to processing batch are captured in the KDD_PRCSNG_BATCH_SRC FSDM table. The following columns are available in this table:

- PRCSNG_BATCH_NM VARCHAR2(20) Not NULL (PK)
- SRC_ORIGIN VARCHAR2(3) Not NULL (PK)
- SRC_DESC VARCHAR2(255)

Processing CTR Data Files

To load CTR data, follow these steps:

1. Execute Group 1 through Group 5 in sequence in the CSA using AAI T2T, or through Flat Files using BDF. For more information, see *Loading T2T using the AAI Framework*.
2. Process the loaded data using BDF datamaps in FSDM. For more information, see *Managing Data Processing*.

Table 19 lists the data files by group.

Table 19. Data Files by Group

Group	Data Files	
1	AccountCustomerRole AccountPhone AccountEmailAddress Organization	
3	Account Customer OrganizationRelationship	
4	AccountToCustomer CustomerAddress CustomerEmailAddress CustomerPhone	AccountToOrgansation BranchCTRConductor BranchCTRTransaction BranchCTRSummary
5	CurrencyTransaction CurrencyTransaction_ExemptFlagUpd	

Processing of data in Group1 requires no prerequisite information (dependencies) for preprocessing. Groups that follow, however, rely on successful preprocessing of the previous group to satisfy any dependencies. For example, Ingestion does not run Group 4 until processing of data in Group 3 completes successfully.

Processing bases the dependencies that determine grouping on the referential relationships within the data. If an Oracle client chooses not to perform referential integrity checking, grouping is not required (except in some instances). In this case, a need still exists to process some reference data files prior to processing trading data. The following sections describe these dependencies.

BDF.xml File Parameters

The following table describes the parameters which must be configured in the BDF.xml file under the `<OFSBDF Installed Directory>/bdf/config` folder.

Property Name	Description	Default
DIS.Source	Indicates the source of DIS records. Valid values are: ● FILE for a DIS file ● FSDW for CSA table loading ● FILE-EXT for loading DIS file using an external table	FILE
DIS.ArchiveFlag	Indicates whether a DIS file should be archived after it has been processed.	true
DIS.BufferSize	Indicates the size of a byte buffer (in kilobytes) used to read in a line from a DIS file. This should be set to the maximum possible record size (in kilobytes) of a record in a DIS file.	100
DIS.InputFileCharset	Indicates the character set of a DIS file.	UTF8
DIS.Default.Check.Requirement	Indicates whether the mandatory and conditional checks on a DIS record should be done	true

Property Name	Description	Default
DIS.Default.Reject.Requirement	Indicates whether a mandatory or conditional check failure for a record should result in the record being rejected. If this is set to FALSE and a missing value is attempted to be inserted into a NOT NULL column, then the record will be rejected anyway.	true
DIS.Default.Check.Domain	Indicates whether the domain value checks on a DIS record should be done.	true
DIS.Default.Reject.Domain	Indicates whether a domain value check failure for a record should result in the record being rejected.	true
DIS.Default.Check.Length	Indicates whether the maximum length checks on a DIS record should be done.	true
DIS.Default.Reject.Length	Indicates whether a maximum length check failure for a record should result in the record being rejected. If this is set to FALSE, then the value will be truncated based on the maximum length of the field.	true
DIS.Default.Check.Threshold	Indicates whether the threshold checks (GREATER_THAN_ZERO, etc) on a DIS record should be done.	true
DIS.Default.Reject.Threshold	Indicates whether a threshold check failure for a record should result in the record being rejected.	true
DIS.Default.Check.Lookup	Indicates whether the reference data lookups on a DIS record should be done.	true
DIS.Default.Reject.Lookup	Indicates whether a reference data lookup failure for a record should result in the record being rejected.	true
MITrxnProducttypes	Indicates the parameter which is used to pass a list of product codes for trailing digit purpose (AUG_INSTR_NB derivation).	● CHECK ● CHECK-ACH
CustProfileLookBack	Indicates the parameter which is used to look back at the days in Customer Summary Daily for Customer Summary Month recalculation. **Note:** In order to look back at a specific time period in Customer Summary Daily, you must have partitions available in Customer Summary Month.	31
CustAcctHolderType	Indicates the parameter which is used to identify customer account types to be included in customer summary.	CI

BDF Directory Structure

The BDF Datamap component is organized as subdirectories below the `<OFSAAI Installed Directory>/bdf` file. The following table provides details about each subdirectory..

Table 20. Directory Structure Description

Directory Name	Description
scripts	Shell scripts for running BDF components, setting the environment, and changing passwords
logs	Log files containing status and error messages produced by BDF components
config	Files used to configure BDF components
config/datamaps	XML files containing data map definitions for individual BDF components
jars	Java Archive (JAR) files used to run BDF components
data/errors	Files containing error records produced by BDF components
data/temp	Temporary files produced by BDF components
inbox	Data files provided by the Oracle client in DIS format
fuzzy_match	C++ library files used for the purpose of fuzzy matching names

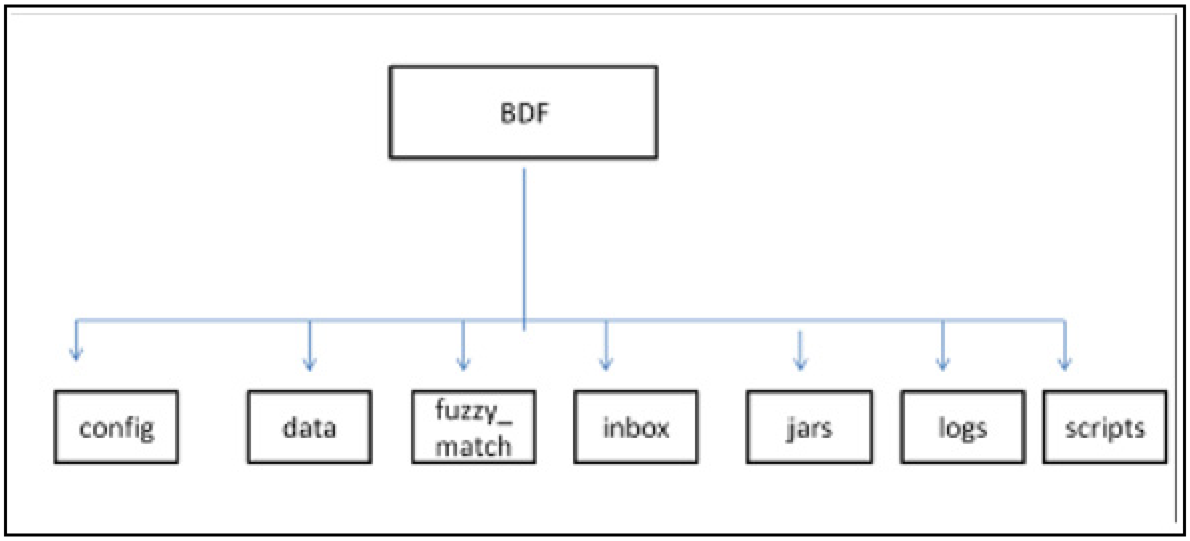


Figure 9. BDF Subsystem Directory Structure

The following sections describe the BDF directory structure.

Scripts

The scripts folder contains the following files:

- **changePassword.sh -** Changes passwords used during the execution of BDF components. Refer to the *Installation Guide* for more information.
- **env.sh -** Sets ups the shell environment of BDF components
- **execute.sh -** Executes BDF components.

For Example:

```
<OFSAAI Installed Directory>/bdf/scripts/execute.sh <component>
<OFSAAI Installed Directory>/bdf/scripts/execute.sh CorrespondentBankProfile
```

Note: *Component* in this document means a batch process which is part of the BDF Datamap subsystem. For the most part, these components will refer to XML data maps. For example, the AccountProfile_Balance component refers to the AccountProfile_Balance.xml data map.

Running these files in the BDF subsystem improves performance time.

Logs

The log file has information about the warnings, errors, and status of the component. Additional information can be obtained from a component by turning on diagnostic logging. This can be done by setting the `Log.DIAGNOSTIC.Enabled` parameter to true. In a production environment, this should be left as false and only changed to true when debugging errors or performance issues.

Log files for each component are written to a log file named for the component inside a subdirectory of the logs directory named for the current processing date in YYYYMMDD format:

For example:

```
<OFSAAI Installed Directory>/bdf/logs/<processing date>/<component>.log
<OFSAAI Installed Directory>/bdf/logs/20130313/CorrespondentBankProfile.log
```

When SQL*Loader is the loading mechanism, as shown below, there are additional log files containing log output from the SQL*Loader utility named the same as the component's log file with "_N" extensions (where **N** is an integer).

For example:

```
<OFSAAI Installed Directory>/bdf/logs/20130313/CorrespondentBankProfile_0.log
<OFSAAI Installed Directory>/bdf/logs/20130313/CorrespondentBankProfile_1.log
```

When an external table is used as the DIS file loading mechanism, there are additional log files containing log output from the external table utility. The log files are named the same as the external table being loaded. The name of the external table is the name of the table being loaded with a prefix of "DIS_". For example, when loading the ACCT table, the external table log file will be:

```
<OFSAAI Installed Directory>/bdf/logs/20130313/DIS_ACCT.log
```

Parameters

Parameters in BDF Datamaps are specified as elements in an XML file. The XSD containing a description of these elements can be found in the following directory:

```
<OFSAAI Installed Directory>/bdf/config/ParameterSet.xsd
```

The Parameter element defines a parameter and its value, and contains the following attributes:

- **name -** The name of the parameter.

- **type -** The data type of the parameter. Valid values are STRING, REAL, INTEGER, BOOLEAN, FILE, and CLASS.
- **value -** The value of the parameter, which must map the type of the parameter.
- **list -** A boolean value specifying that the value is a single value (false - the default) or a comma separated list of values (true).

For example:

```
<Parameter name="MinimumGeographyRisk" type="INTEGER" value="0"/>
<Parameter name="InternalAccountCodeList" type="STRING" value="IA,GL" list="true"/>
```

Note: If the value of the parameter is a string containing characters which are not allowed in an XML attribute, then a CDATA element can be used as the element's text.
For example:

```
<Parameter name="PassThruExpressionSeparators" type="STRING">
<![CDATA[~: \t/#-]]>
</Parameter>
```

Parameters in the main BDF.xml file should not be modified. Instead, any customizations to parameter values should be placed in the <OFSAAI Installed Directory>/bdf/config/custom/BDF.xml file. Parameters can be overridden at the component level by placing them in the custom/<component>.xml file. Also, parameters can be overridden on the command line by passing the parameter name and value as parameters to the execute.sh script after the component name:

For example:

```
<OFSAAI Installed Directory>/bdf/scripts/execute.sh <component> [parameter name=value]*
<OFSAAI Installed Directory>/bdf/scripts/execute.sh CorrespondentBankProfile
NumberOfThreads=4
```

When a given parameter is read by a component, the order of precedence for where the parameter value is taken from is as follows:

```
    command line
<OFSAAI Installed Directory>/bdf/config/custom/<component>.xml
<OFSAAI Installed Directory>/bdf/config/<component>.xml
<OFSAAI Installed Directory>/bdf/config/custom/BDF.xml
<OFSAAI Installed Directory>/bdf/config/BDF.xml
```

Config

The config subdirectory contains configuration files.

- `<OFSAAI Installed Directory>/bdf/config/BDF.xml` contains all default product configuration parameters. It should not be modified.
- `<OFSAAI Installed Directory>/bdf/config/install/BDF.xml` contains all configuration parameters set at installation time (refer to the *Installation Guide* for more information).

- `<OFSAAI Installed Directory>/bdf/config/custom/BDF.xml` contains any product configuration parameters that have been overridden for this installation. It is initially empty. Any changes to default product configuration parameters should be put here.

Individual BDF components can have their own configuration file which overrides default product parameters. These files would be named using the following format:

`<OFSAAI Installed Directory>/bdf/config/<component>.xml`

For example:

`<OFSAAI Installed Directory>/bdf/config/CorrespondentBankProfile.xml`

Component configuration files in this directory are part of the product and should not be modified. If any parameters must be overridden at the individual component level, the component configuration file should be created in `<OFSAAI Installed Directory>/bdf/config/custom`.

- The datamaps subdirectory contains XML files holding the data map definitions for BDF components.
- The derivations subdirectory contains SQL derivations for individual fields.
- The queries subdirectory contains SQL queries for individual data maps.

BDF.xml Configuration Parameters

The following table describes the BDF properties configurations mentioned in the `<OFSAAI Installed Directory>/bdf/config/BDF.xml` file.

Table 21. BDF.xml File Configuration Parameters

Parameter Name	Description	Example
MISCELLANEOUS		
NumberOfThreads	The number of worker threads used by some BDF components	4
SequenceBatchSize	The batch size when retrieving sequence IDs for new records	100000
SourceSystem	he default value for source system when one is not provided	MTS
Currency	The default value for issuing currency when one is not provided	USD
Separator	The delimiter that separates fields in data file records.	~
DB: Parameters related to database access.		
DB.Connection.Driver	The JDBC driver class name.	oracle.jdbc.OracleDriver
DB.Timeout	The number of seconds to wait before timing out on a database connection attempt.	10
DB.NumRetries	The maximum number of times to attempt to connect to a database before failing.	5
DB.MaxNumberOfDeadlocks	The maximum number of times a deadlock is encountered during a JDBC insert or update operation, before an error is generated.	10
Directory: Parameters used to define directory locations.		
Directory.Inbox	The input directory where the Oracle client will write DIS files. Date subdirectories will be created in this directory where these files will be archived	../inbox

Table 21. BDF.xml File Configuration Parameters

Parameter Name	Description	Example
Directory.InternalData	The directory where files generated by BDF components will reside. This includes log files, error files, and any temporary processing files.	..
Log: Parameters used to configure the common logging module		
Log.Format	Identifies the log formatting string.	%d [%t] %p - %m%n
Log.UseDefaultLog	Specifies whether the system uses the default log file for a component. The default log file has the name of the component and resides in a date subdirectory of the logs directory (in YYYYMMDD format).	true
Log.SysLogHostName	The host name of syslog for messages sent to syslog.	hostname
Log.SMTPHostName	The host name of the SMTP server for messages that processing sends to an e-mail address.	hostname
Log.MaxSize	The maximum size (in MB) of a log file before the system creates a new log file.	2000MB
Log.MaxIndex	If a log file exceeds Log.MaxSize, this will be the maximum number of additional log files that are created (Component.log.1, Component.log.2, etc).	10
Log.TRACE.Enabled	Indicates that trace logging is not enabled; true indicates enabling of trace logging.	false
Log.TRACE.Location	Specifies additional locations to send TRACE log messages to, other than the default BDF log file (logs/YYYYMMDD/Component.log). If the value is not provided, considers the default BDF log location.	false
Log.TRACE.Synchronous	Specify whether logging for a particular level should be performed synchronously or asynchronously.	false
Log.DIAGNOSTIC.Enabled	DIAGNOSTIC logging is used to log database statements and will slow down performance. Make it true if needed.	false
Log.DIAGNOSTIC.Location	Additional locations to send DIAGNOSTIC log messages to, other than the default BDF log file (logs/YYYYMMDD/Component.log). If the value is not provided, considers the default BDF log location.	
Log.DIAGNOSTIC.Synchronous	Specify whether logging for a particular level should be performed synchronously or asynchronously.	false
Log.NOTICE.Enabled	Indicates enabling of notice logging; false indicates that notice logging is not enabled.	true
Log.NOTICE.Location	Specifies additional locations to send NOTICE log messages to, other than the default BDF log file (logs/YYYYMMDD/Component.log). If the value is not provided, considers the default BDF log location.	
Log.NOTICE.Synchronous	Specify whether logging for a particular level should be performed synchronously or asynchronously.	false
Log.WARN.Enabled	Indicates enabling of warning logging; false indicates that warning logging is not enabled.	true
Log.WARN.Location	Specifies additional locations to send WARN log messages to, other than the default BDF log file (logs/YYYYMMDD/Component.log).	
Log.WARN.Synchronous	Specify whether logging for a particular level should be performed synchronously or asynchronously.	false

Table 21. BDF.xml File Configuration Parameters

Parameter Name	Description	Example
Log.FATAL.Enabled	Indicates enabling of Fatal logging; false indicates that fatal logging is not enabled.	true
Log.FATAL.Location	Specifies additional locations to send FATAL log messages to, other than the default BDF log file (logs/YYYYMMDD/Component.log).	
Log.FATAL.Synchronous	Specify whether logging for a particular level should be performed synchronously or asynchronously.	false
Load: Parameters used to configure common Loading data		
Load.FullRefresh	For DIS files defined as Overwrite, whether to fully replace FSDM tables with the contents of the DIS file (true) or to treat the DIS file as a delta (false)	True
Load.BatchSize	The batch size when loading data.	5000
Load.Direct	Specifies whether to use direct path loading (TRUE) or conventional path loading (FALSE).	false
Load.Unrecoverable	Specifies whether a direct path load does not use redo logs (TRUE) or uses redo logs (FALSE).	false
Load.Partitioned	Specifies whether a direct path load uses the current date partition (TRUE) or any partition (FALSE).	false
Load.SkipIndexes	Specifies whether a direct path load skips index maintenance (TRUE) or maintains indexes (FALSE). If set to TRUE, rebuilding of indexes must occur after running the DataMap XML.	false
Load.DoAnalyze	Specifies whether to run a stored procedure to analyze a database table after loading data into it.	true
Load.AnalyzeType	Specifies the type of analyze statistics has to perform if DoAnalyze has a value of True.	DLY_POST_ LOAD
Load.LogRecordInterval	Specifies how often to log a message saying how many records a particular thread has inserted/updated,	1000
Load.MaxErrorRate	Specifies the percentage of invalid records to allow before exiting with an error. For example, a value of 10 allows 10 percent of records to be invalid before exiting with an error. A value of 0 allows no invalid records. A value of 100 allows all invalid records.	100
Load.RecordQueueSize	Specifies the number of records the query reader thread will write to a database writer thread queue before waiting for the reader thread to catch up. Higher values will require more memory usage.	100
Load.SkipIndexesErrorCode	Specifies a database error code that occurs in the log file when skipping index maintenance.	26025
Load.IndexParallelLevel	Specifies the parallel level of an index rebuild (that is, number of concurrent threads for rebuilding an index).	1
Load.DataErrorCodes	Specifies a comma-separated list of database error codes that indicate data level errors , such as data type and referential integrity. This results in rejection of records with a warning instead of a fatal failure.	1,1400,1401, 1407,1438,1722,1840,1841,2291,2359,1839,1847,12899
Load.ParallelLevel	Specifies the level of parallelization to apply when loading data from a set of source tables to a target table.	8
Load.WriteErrorFiles	Whether to check a DIS file for errors before loading as an external table (true) or not (false)	True

Table 21. BDF.xml File Configuration Parameters

Parameter Name	**Description**	**Example**
DIS: Parameters related to processing DIS files		
DIS.Source	The mechanism used to load DIS data. **FILE:** DIS files will be provided and will be loaded using SQL*Loader processes running on the application server. **FILE-EXT:** DIS files will be provided and will be loaded using external tables with the DIS files accessed directly by the database. **FSDW:** DIS data will be obtained from database tables in the FSDW.	FILE
DIS.ArchiveFlag	Whether DIS files will be archived to a date subdirectory (true) or not (false).	True
DIS.BufferSize	The size in KB of the byte buffer used to read in DIS file records.	100
DIS.InputFileCharset	The character set of the DIS files. Note that output data is always written in UTF8, this parameter just allows the DIS files to be in a different character set.	
DIS.Default.Check.Requireme nt	Whether to check for mandatory fields on DIS records (true) or not (false).	True
DIS.Default.Reject.Requireme nt	Whether to reject DIS records for failing a mandatory field check (true) or to log a warning and attempt to load the record (false).	True
DIS.Default.Check.Domain	Whether to check that a DIS field has a valid domain value (true) or not (false).	True
DIS.Default.Reject.Domain	Whether to reject DIS records that fail a domain check (true) or not (false).	True
DIS.Default.Check.Length	Whether a DIS field should be checked for a valid length (true) or not (false).	True
DIS.Default.Reject.Length	Whether to reject DIS records that fail a length check (true) or not (false)	True
DIS.Default.Check.Threshold	Whether a DIS field should be checked that it is within an acceptable threshold (i.e. greater than 0) (true) or not (false).	True
DIS.Default.Reject.Threshold	Whether to reject DIS records that fail a threshold check (true) or not (false).	True
DIS.Default.Check.Lookup	Not currently supported.	True
DIS.Default.Reject.Lookup -	Not currently supported	True
Parameters used by queries defined in the data maps:		
MinimumGeographyRisk	Defines what is considered High Risk For the Account Profile attributes related to High Risk Geography , such as Incoming High Risk Wire Count. Processing compares this parameter using a strict greater-than operation.	0
AccountInactivityInMonths	Specifies the number of months that processing aggregated to determine whether an account is inactive. If the sum of trades and transactions over this number of months is <= 3, the account is considered inactive. This setting can impact the Escalation in Inactive Accounts scenario. The default value is six months.	6

Table 21. BDF.xml File Configuration Parameters

Parameter Name	Description	Example
TransactionsReversalLookbackDays	This parameter controls how many days of transactions to look across. Verify whether the new data contains reversals of prior transactions.	7
LowPriceSecurityThreshold	Defines Low Priced in the base currency for the Account Profile attributes named Low-Priced Equity Range # Opening Trade Count. Processing compares the value of this parameter to the Trade table's Last Execution Price-Base.	5000
CommissionEquityPercentUpperLimit	Defines the upper limit for Commission Versus Average Daily Equity Percentage in Account Profile Calculation.	5
TurnOverRateUpperLimit	Defines the upper limit for Total Turnover Rate in Account Profile Calculation.	5
BankCodeListWithIA	Defines the List of Financial Institution Identifier Types, these are type of unique identifiers which are used to represent the financial institutions. This parameter also contains IA (Internal Account Identifier) to be used in datamaps and is mainly used in Correspondent Bank related datamap derivations. Below are the list of examples ● BIC: SWIFT Bank Identifier Code (BIC) ● CHU: CHIPS Participant User Identifier ● CO: Corporate Identifier ● CHP: CHIPS Participant Identifier ● FED: Federal Reserve Routing (ABA) Number ● CU: Customer Identifier ● GL: General Ledger Account ● IA: Internal Account Identifier	BIC,FED,CHP,CHU, DTC,CDL,EPN,KID, CBI,CSN,OTF,BLZ,IBAN,ABLZ,BSB,CP AP, SDIC, HEBIC, BCHH, NSC, IFSC, IDIC, PNCC, RCBIC, UKDSC, Swiss BC, Swiss SIC,IA
BankCodeList	Defines the List of Financial Institution Identifier Types, these are type of unique identifiers which are used to represent the financial institutions excluding Internal Account (IA). This parameter does not contain IA (Internal Account Identifier) to be used in datamaps and is typically used to derive financial institutions. Below are the list of examples ● BIC: SWIFT Bank Identifier Code (BIC) ● CHU: CHIPS Participant User Identifier ● CO: Corporate Identifier ● CHP: CHIPS Participant Identifier ● FED: Federal Reserve Routing (ABA) Number ● CU: Customer Identifier ● GL: General Ledger Account	BIC,FED,CHP,CHU, DTC,CDL,EPN,KID, CBI,CSN,OTF,BLZ,IBAN,ABLZ,BSB,CP AP, SDIC, HEBIC, BCHH, NSC, IFSC, IDIC, PNCC, RCBIC, UKDSC, Swiss BC, Swiss SIC

Table 21. BDF.xml File Configuration Parameters

Parameter Name	Description	Example
IdRiskWinLevel	Defines the Risk level to calculate Effective Risks for internal parties (Account/ Customer). For example: Account 1234 has an Effective Risk of 5, IdRiskWinLevel can be set by the client. If the party identifier effective risk is greater than the set IdRiskWinLevel, then the party identity risk wins compared to fuzzy matcher (Party Name Risk). If not, fuzzy matcher wins.	1
InternalAccountCodeList	Codes to define types of Internal Entities with client, for example: ● IA: Internal Account Identifier ● GL: General Ledger Account	IA, GL
ExternalEntityCodeList	Codes to define types of External Entities with client, for example: ● XA: External Account Identifier ● CO: Corporate Identifier ● DL: Driver License ● IBAN: International Bank Account Number	XA,CC,CO,DL,GM, GP,LE,MC,ND,NR, PP,SS,TX,AR,OT,IB AN
TrustedPairReviewReasonText1	Defines the reason text1 for recommendation of cancelling the Trusted Pair, due to increase in Risk of parties involved in trusted pair.	Risk of <Party1> increased from <A> to <b>
TrustedPairReviewReasonText2	Defines the reason text2 for recommendation of cancelling the Trusted Pair, due to increase in Risk of parties involved in trusted pair.	Risk of <Party2> increased from <C> to <D>
CorporateActionLookBackDays	This parameter determines the how many days trades to look back from the Corporate Effective Date.	7
DealNearTermMaturityDays	Defines the maximum number of days between the End Date and Trade Date. This helps to calculate Structured Deals Initiated w/ Near-Term Exp. In Customer Profile/ Institutional Account Profile.	7
ProfitLossUpperLimit	Helps determine how much a security must move by the end of the day to be considered a win or loss. If the security moves by less than a specified percentage, processing does not count it either way. If it moves by this percentage or more, it counts as a win or a loss, depending on whether the movement was beneficial to the account that made the trade.	5
HouseholdTurnOverRateUpperLimit	Defines the upper limit for Total Turnover Rate in Household Profile Calculation.	10000
HouseholdCommissionEquityPercentUpperLimit	Defines the upper limit for Commission Versus Average Daily Equity Percentage in Account Profile Calculation.	10000
OptionTradeAmountRange1 OptionTradeAmountRange2 OptionTradeAmountRange3 OptionTradeAmountRange4 OptionTradeAmountRange5 OptionTradeAmountRange6	Define the lower bound of each range for the Account Profile attributes named Options Range # Opening Trade Count. Processing compares each parameter to the Trade table's Last Principal Amount- Base. Each range is from the lower bound entered here to the lower bound of the next range.	

Table 21. BDF.xml File Configuration Parameters

Parameter Name	Description	Example
EquityTradeAmountRange1 EquityTradeAmountRange2 EquityTradeAmountRange3 EquityTradeAmountRange4 EquityTradeAmountRange5 EquityTradeAmountRange6	Define the lower bound of each range for the Account Profile attributes named Equity Range # Opening Trade Count. Processing compares each parameter to the Trade table's Last Principal Amount- Base. Each range is from the lower bound entered here to the lower bound of the next range.	
LowPricedEquityTradeAmount Range1 LowPricedEquityTradeAmount Range2 LowPricedEquityTradeAmount Range3 LowPricedEquityTradeAmount Range4 LowPricedEquityTradeAmount Range5 LowPricedEquityTradeAmount Range6	Define the lower bound of each range for the Account Profile attributes named Low-Priced Equity Range # Opening Trade Count. Processing compares each parameter to the Trade table's Last Principal Amount-Base. Each range is from the lower bound entered here to the lower bound of the next range.	
MutualFundTradeAmountRang e1 MutualFundTradeAmountRang e2 MutualFundTradeAmountRang e3 MutualFundTradeAmountRang e4 MutualFundTradeAmountRang e5 MutualFundTradeAmountRang e6	Define the lower bound of each range for the Account Profile attributes named Mutual Fund Range # Opening Trade Count. Processing compares each parameter to the Trade table's Last Principal Amount-Base. Each range is from the lower bound entered here to the lower bound of the next range.	
UnrelatedWhenOffsetAccountI sNull	This parameter is used to assign unrelated party code as "J" in the BackOfficeTransaction table, If OFFST_ACCT_INTRL_ID is null and UnrelatedWhenOffsetAccountIsNull is "Y", If OFFST_ACCT_INTRL_ID is null and UnrelatedWhenOffsetAccountIsNull is "N", then unrelated party code is NULL.	Y

BDF Datamap Configuration File

Oracle clients can modify the BDF.xml file under the bdf/config/custom folder to override default settings that the system provides. You can also reapply any modifications in the current BDF.xml file to the newer BDF.xml file.

Override any settings in BDF.xml by placing the modifications in BDF.xml under the bdf/config/custom folder.

During installation, the following parameters are configured by the installer:

- `AccountTrustFromCustomer`
- `DefaultJurisdiction`
- `UseTaxidForUnrelatedPartyCode`
- `BaseCountry`

- `ProcessForeignFlag`
- `ProcessBankToBank`
- `ProcessTransactionXRefFlag`
- `TrustedPairRiskReviewFlag`

These parameters are stored in the following file:

`<OFSAAI Installed Directory>/bdf/config/install/BDF.xml`

Parameters DefaultJurisdiction and BaseCountry are defined in the InstallConfig.xml file during Silent Installation. Refer to the *Installation Guide* for more information.

The Installer sets the default value for other parameters as follows:

- `<Parameter name="AccountTrustFromCustomer" type="STRING" value="Y"/>`
- `<Parameter name="DefaultJurisdiction" type="STRING" value="AMEA"/>`
- `<Parameter name="UseTaxidForUnrelatedPartyCode" type="STRING" value="Y"/>`
- `<Parameter name="BaseCountry" type="STRING" value="US"/>`
- `<Parameter name="ProcessForeignFlag" type="STRING" value="N"/>`
- `<Parameter name="ProcessBankToBank" type="STRING" value="N"/>`
- `<Parameter name="ProcessTransactionXRefFlag" type="STRING" value="Y"/>`
- `<Parameter name="TrustedPairRiskReviewFlag" type="STRING" value="N"/>`

To change the default value of these parameters, before running ingestion, go to `<OFSAAI Installed Directory>/bdf/config/install/BDF.xml` and change the value to 'Y' or 'N' as needed.

The following table describes the parameters defined in BDF.xml:

Table 22. BDF Datamap Configuration Parameters

Property Name	Description	Example
DB.Connection.URL	Database URL for JDBC connections made by BDF components. The content and format of this value is specific to the database vendor and the vendor database driver.	jdbc:oracle:thin:@solitaire.mantas.com:1521:D1O9L2
DB.Connection.Instance	Database instance to connect to on the database servers. Typically, the instance name matches the database name portion of the DB.Connection.URL.	D1O9L2
DB.Connection.Password	Password that Java Ingestion components use when connecting with the database. This is set by executing bdf/scripts/changepassword.sh	
DB.Schema.MANTAS	Schema name for the Oracle ATOMIC database schema. BDF accesses the ATOMIC schema when allocating sequence IDs to ingested records.	ATOMIC
DB.Schema.MARKET	Schema name for the ATOMIC database schema. Data Management stores market data related records in the ATOMIC schema.	ATOMIC
DB.Schema.BUSINESS	Schema name for the ATOMIC database schema. Data Management stores business data related records in the ATOMIC schema.	ATOMIC
DB.Schema.CONFIG	Name of the configuration schema owner.	REVELEUS
DB.Schema.CASE	Name of the ATOMIC schema owner.	ATOMIC

Table 22. BDF Datamap Configuration Parameters

Property Name	Description	Example
DB.Alg.Connection.User	Database user for running Behavior Detection post-processing jobs.	ATOMIC
DB.Alg.Connection.Password	Password for the DB.Alg.Connection.User.	

There are also configuration files for individual components that are delivered as part of the product as:

`<OFSAAI Installed Directory>/bdf/config/<component>.xml`

And can also be created in the following:

`<OFSAAI Installed Directory>/bdf/config/custom/<component>.xml`

Alternatives to Standard Data Management Practices

Data Management Archiving

During ingestion processing, the system moves processed files into an archive directory. Firms can use these files to recover from processing malfunctions, and they can copy these files to off-line media for backup purposes.

The preprocessor moves files in the `/inbox` directory. All other components move their input files to date-labeled subdirectories within the `/backup` directory.

Periodically, an Oracle client can run the `runIMC.sh` script to perform the Data Management cleanup activities. This script deletes old files from the archive area based on a configurable retention date. Periodic running of the cleanup script ensures that archive space is available to archive more recent data.

Archiving Database Information

The Data Ingestion subsystem uses the following procedure:

1. Processing places data in the newest partition of the partitioned tables.
2. Scenarios examine the data in the partitioned tables; the system then generates alerts for detected behaviors.
3. Historical Data Copy processing copies the information that generated alerts reference to the *_ARC* archive tables. The Platform UI displays alert information from the archive tables and information from the non-archived tables. This ensures that the alert information is in the same state as when the system generated the alert, while the most recent information is available to the user.

CHAPTER 5 Post-Processing Tasks

This chapter defines the following post-processing administrative tasks:

- About Post-Processing
- Alert Creation
- Batch Execution of CTR
- About CTR Web Service Invocation

About Post-Processing

During post-processing of ingested data, Behavior Detection prepares the detection results for presentation to users. Preparation of the results depends upon the following processes:

- **Augmentation:** Collects information for pattern detection, which enables proper display or analysis of these results may be required

Note: The Match Augmentation process is no longer explicitly run as a separate job. It is automatically executed at the end of each scenario run.

- **Alert Creation:** Packages the scenario matches as units of work (that is, alerts), potentially grouping similar matches together, for disposition by end users
- **Batch Execution of CTR**: The CTR Batch should be executed every day after alert creation job is run.

Order of Running Post-Processing Administrative Tasks

Run the post-processing administrative tasks in this order:

1. Alert Creation (503)
2. Batch Execution of CTR

Alert Creation

Matches are converted into alerts with the Alert Creator processes. The system uses match alert creator job to generate one alert per match

Running the Alert Creation Job

The Alert Creator is part of the Behavior Detection subsystem. Behavior Detection provides default job templates and job template groups for running Alert Creator.

The following section describes how to run the match alert creator.

To Run Match Alert Creator

To run the match Alert Creator, follow these steps:

1. Verify that the dispatcher is running.
2. Run the `start_mantas.sh` script as follows:

   ```
   start_mantas.sh 503
   ```

   where `503` is the job template that Behavior Detection provides to run the Alert Creator algorithm.

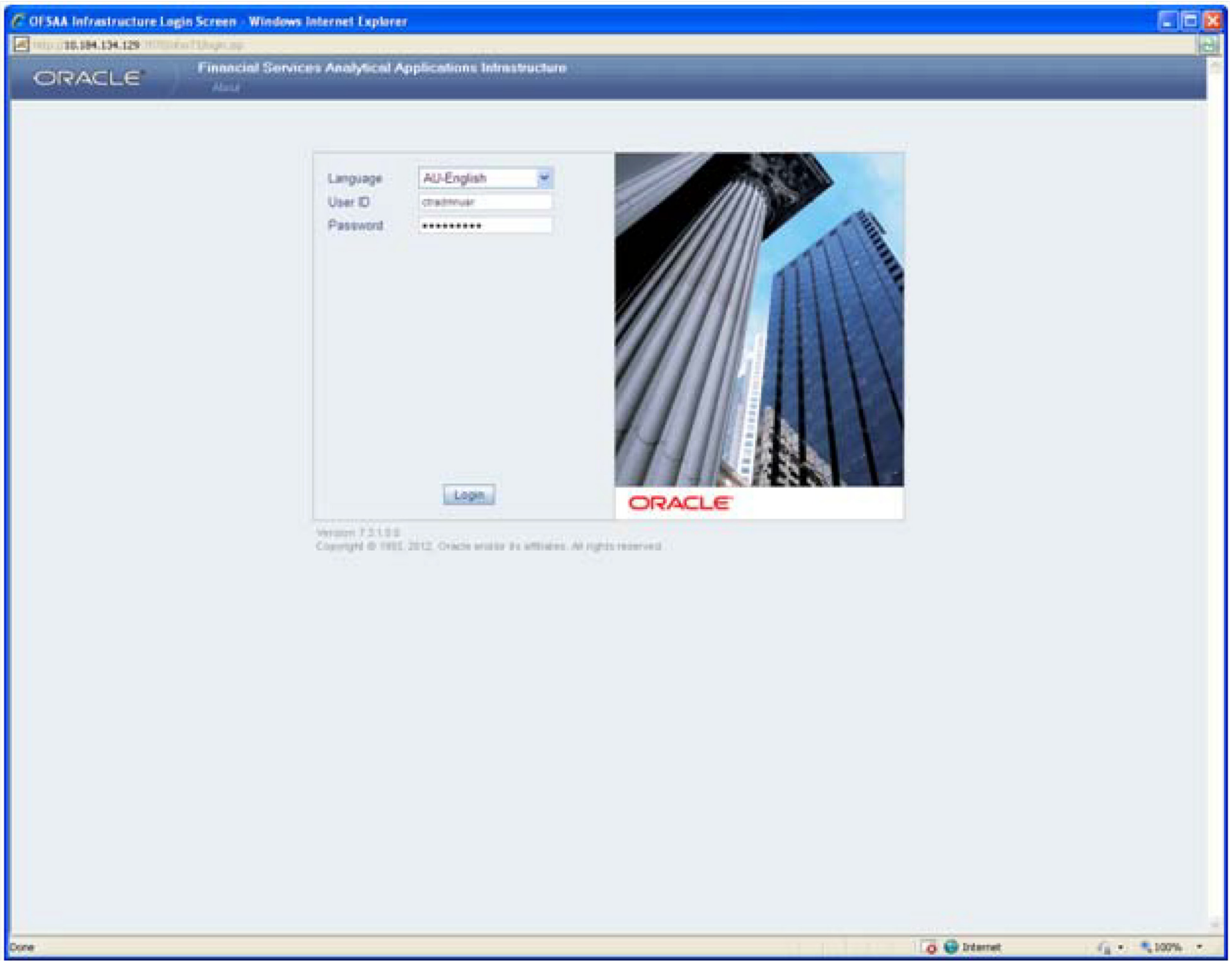
Batch Execution of CTR

The CTR Batch should be executed every day after post processing of all the alerts is done.

Note: Analyze your tables before executing the CTR Batch. See *Database Statistics Management,* for more information.

Follow these steps to execute the OFSAAI batch for CTR (Create CTR):

1. Login to OFSAAI as a CTR Administrator User.

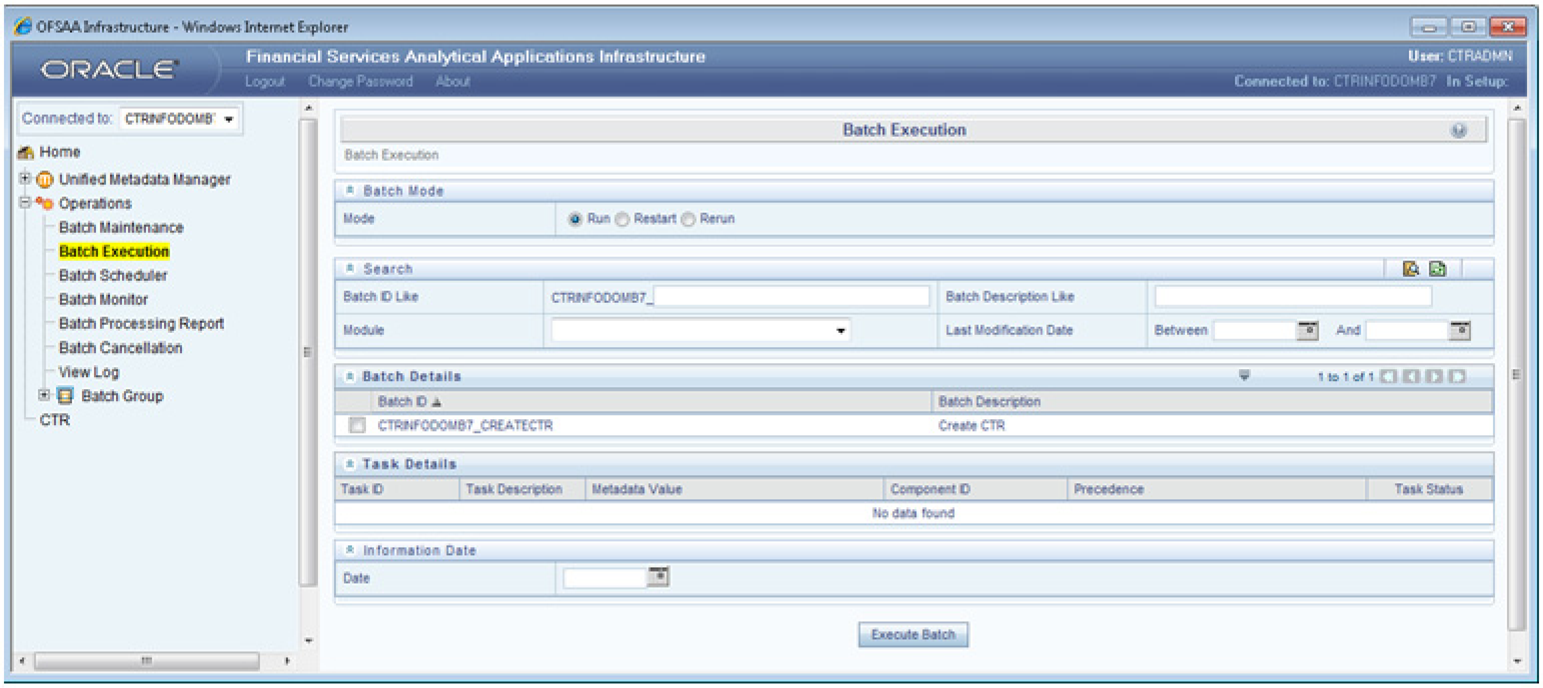

OFSAA Infrastructure Login Screen - Windows Internet Explorer
ORACLE
Financial Services Analytical Applications Infrastructure
About
Login
ORACLE
Done
Internet
100%

Language: AU-English
User ID: ctradmnuser
Password: **********

Version 7.3.1.0.0
Copyright © 1993, 2012, Oracle and/or its affiliates. All rights reserved.

Figure 10. OFSAAI Login Screen

2. Select the **CTR** infodom.
3. Click **Operations** to expand the LHS menu, and then click **Batch Execution**.


OFSAA Infrastructure - Windows Internet Explorer

ORACLE — Financial Services Analytical Applications Infrastructure — User: CTRADMN
Logout Change Password About — Connected to: CTRINFODOMB7 In Setup:

Connected to: CTRINFODOMB7

- Home
- Unified Metadata Manager
- Operations
  - Batch Maintenance
  - Batch Execution
  - Batch Scheduler
  - Batch Monitor
  - Batch Processing Report
  - Batch Cancellation
  - View Log
  - Batch Group
- CTR

Batch Execution

Batch Execution

Batch Mode

Mode	[● Run] ○ Restart ○ Rerun

Search

Batch ID Like	CTRINFODOMB7_	Batch Description Like	
Module		Last Modification Date	Between … And …

Batch Details — 1 to 1 of 1

	Batch ID	Batch Description
☐	CTRINFODOMB7_CREATECTR	Create CTR

Task Details

Task ID	Task Description	Metadata Value	Component ID	Precedence	Task Status
No data found					

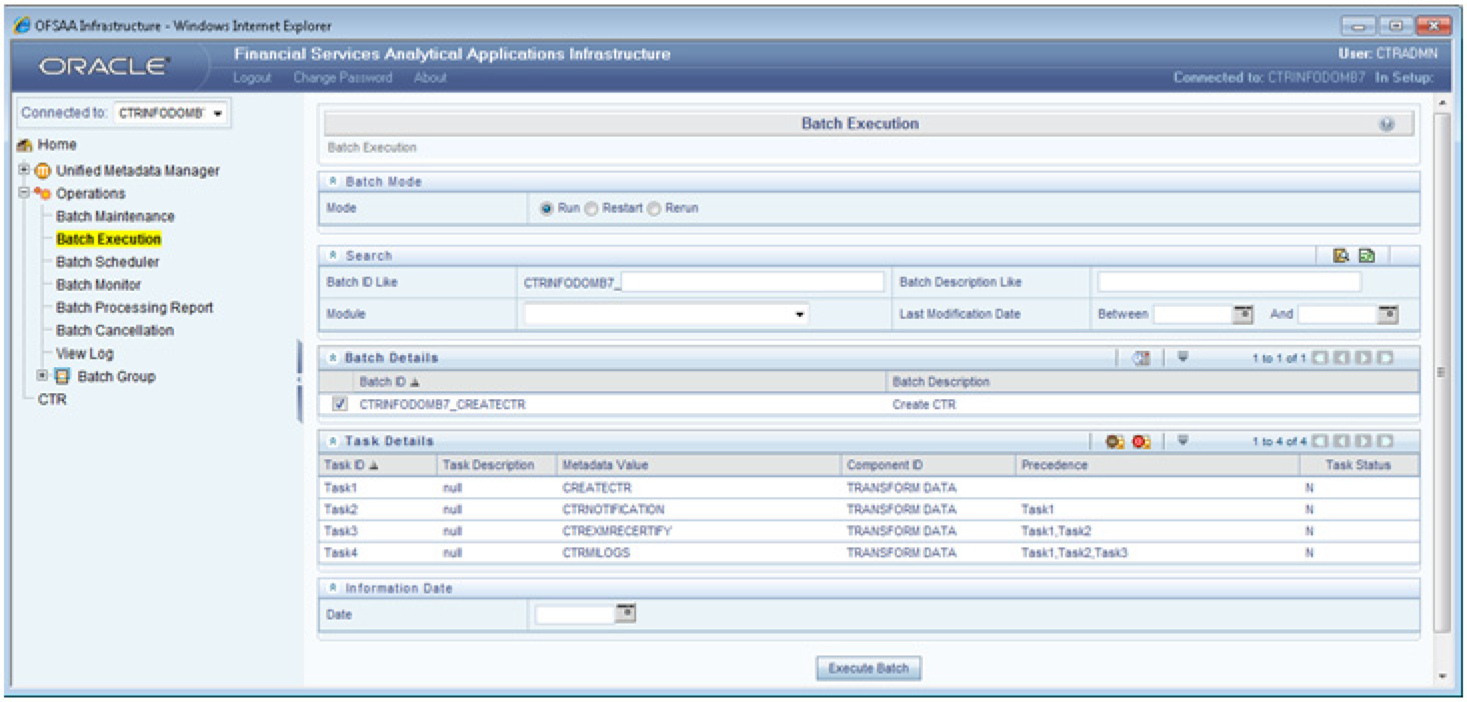
Information Date

Date	

Execute Batch

Figure 11. CTR Batch Execution Link

4. Select the batch (Create CTR) from the Batch Details section.


OFSAA Infrastructure - Windows Internet Explorer

ORACLE — Financial Services Analytical Applications Infrastructure — User: CTRADMN
Logout Change Password About — Connected to: CTRINFODOMB7 In Setup:

Connected to: CTRINFODOMB7

- Home
- Unified Metadata Manager
- Operations
  - Batch Maintenance
  - Batch Execution
  - Batch Scheduler
  - Batch Monitor
  - Batch Processing Report
  - Batch Cancellation
  - View Log
  - Batch Group
- CTR

Batch Execution

Batch Execution

Batch Mode

Mode	[● Run] ○ Restart ○ Rerun

Search

Batch ID Like	CTRINFODOMB7_	Batch Description Like	
Module		Last Modification Date	Between … And …

Batch Details — 1 to 1 of 1

	Batch ID	Batch Description
☑	CTRINFODOMB7_CREATECTR	Create CTR

Task Details — 1 to 4 of 4

Task ID	Task Description	Metadata Value	Component ID	Precedence	Task Status
Task1	null	CREATECTR	TRANSFORM DATA		N
Task2	null	CTRNOTIFICATION	TRANSFORM DATA	Task1	N
Task3	null	CTREXMRECERTIFY	TRANSFORM DATA	Task1,Task2	N
Task4	null	CTRMLOGS	TRANSFORM DATA	Task1,Task2,Task3	N

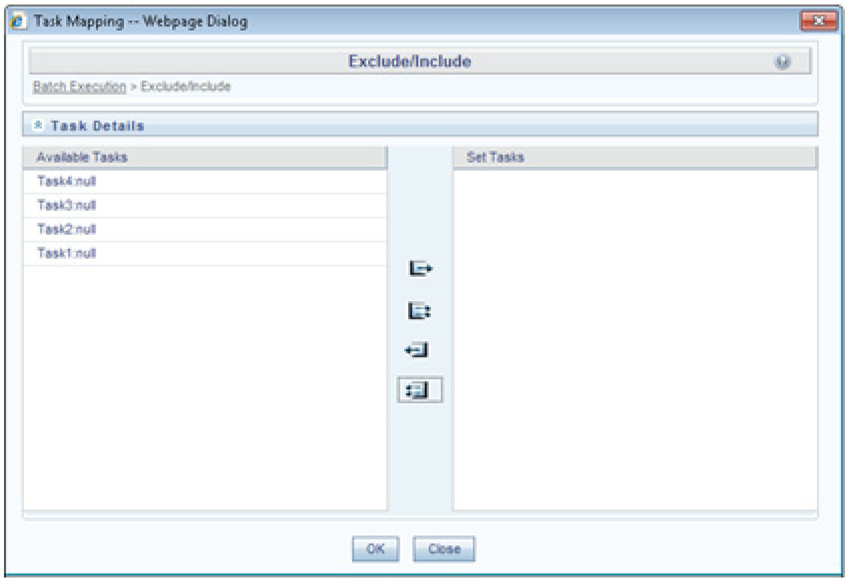
Information Date

Date	

Execute Batch

Figure 12. CTR Batch Execution - Select Batch ID

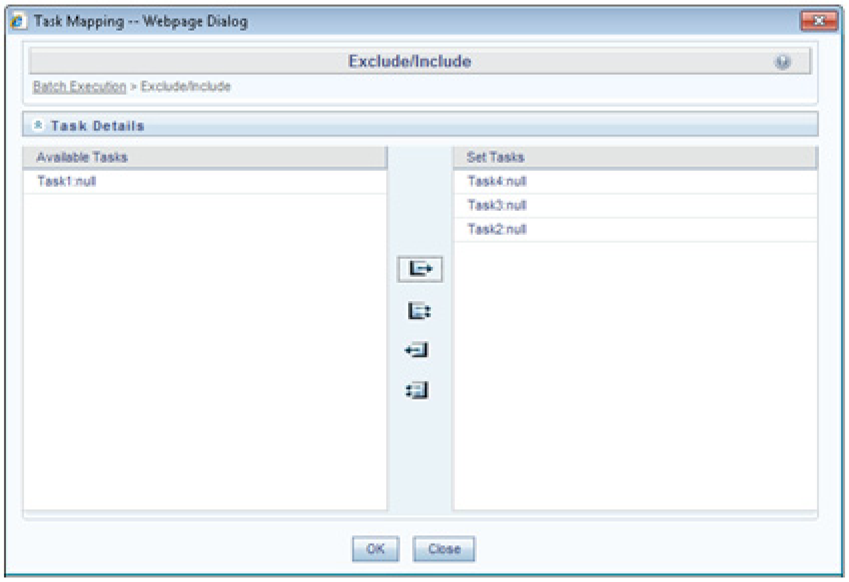
5. Click the **Exclude/Include** icon in the Task details. The Task Mapping window displays.


Task Mapping -- Webpage Dialog
Exclude/Include
Batch Execution > Exclude/Include
Task Details
Available Tasks
Set Tasks
OK
Close

Task4:null
Task3:null
Task2:null
Task1:null

Figure 13. Test Mapping Window

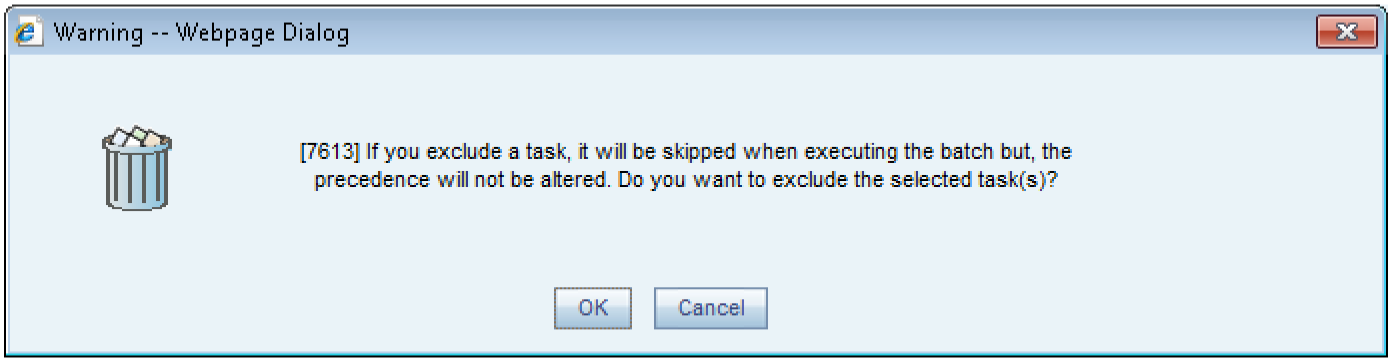
6. Keep the required tasks that you want to execute under the Available Tasks and move the rest to the Set Tasks section.


Task Mapping -- Webpage Dialog
Exclude/Include
Batch Execution > Exclude/Include
Task Details
Available Tasks
Set Tasks
OK
Close

Available Tasks	Set Tasks
Task1:null	Task4:null
	Task3:null
	Task2:null

Figure 14. Test Mapping Window - Select Tasks

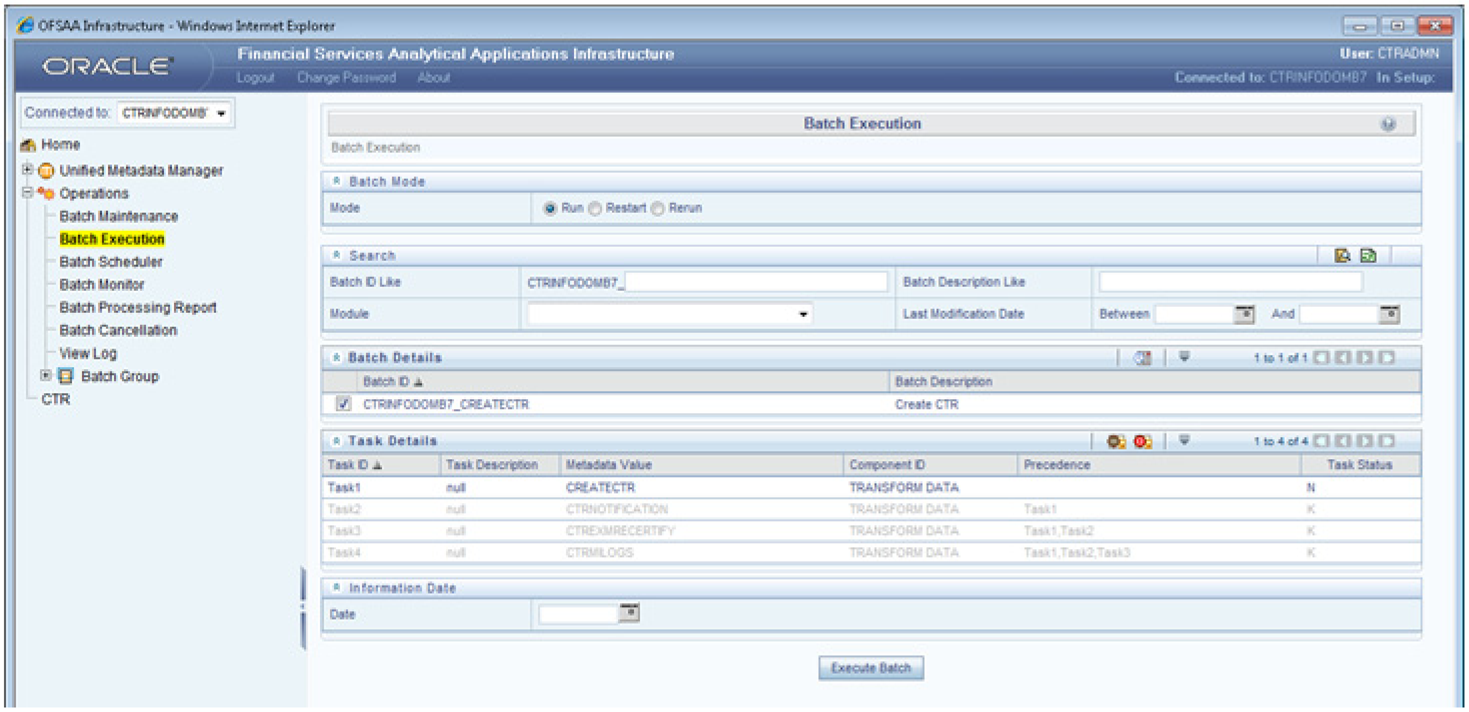
7. Click **OK**. A warning message dispalys:



[7613] If you exclude a task, it will be skipped when executing the batch but, the precedence will not be altered. Do you want to exclude the selected task(s)?

Figure 15. Warning Message

8. Click **OK**.The Batch Execution screen is displayed with the selected tasks to be executed. The selected tasks to be executed are highlighted.



Connected to: CTRINFODOMB7

Batch Mode

Mode: (● Run) ○ Restart ○ Rerun

Search

Batch ID Like: CTRINFODOMB7_
Batch Description Like:
Module:
Last Modification Date: Between … And …

Batch Details (1 to 1 of 1)

	Batch ID	Batch Description
✓	CTRINFODOMB7_CREATECTR	Create CTR

Task Details (1 to 4 of 4)

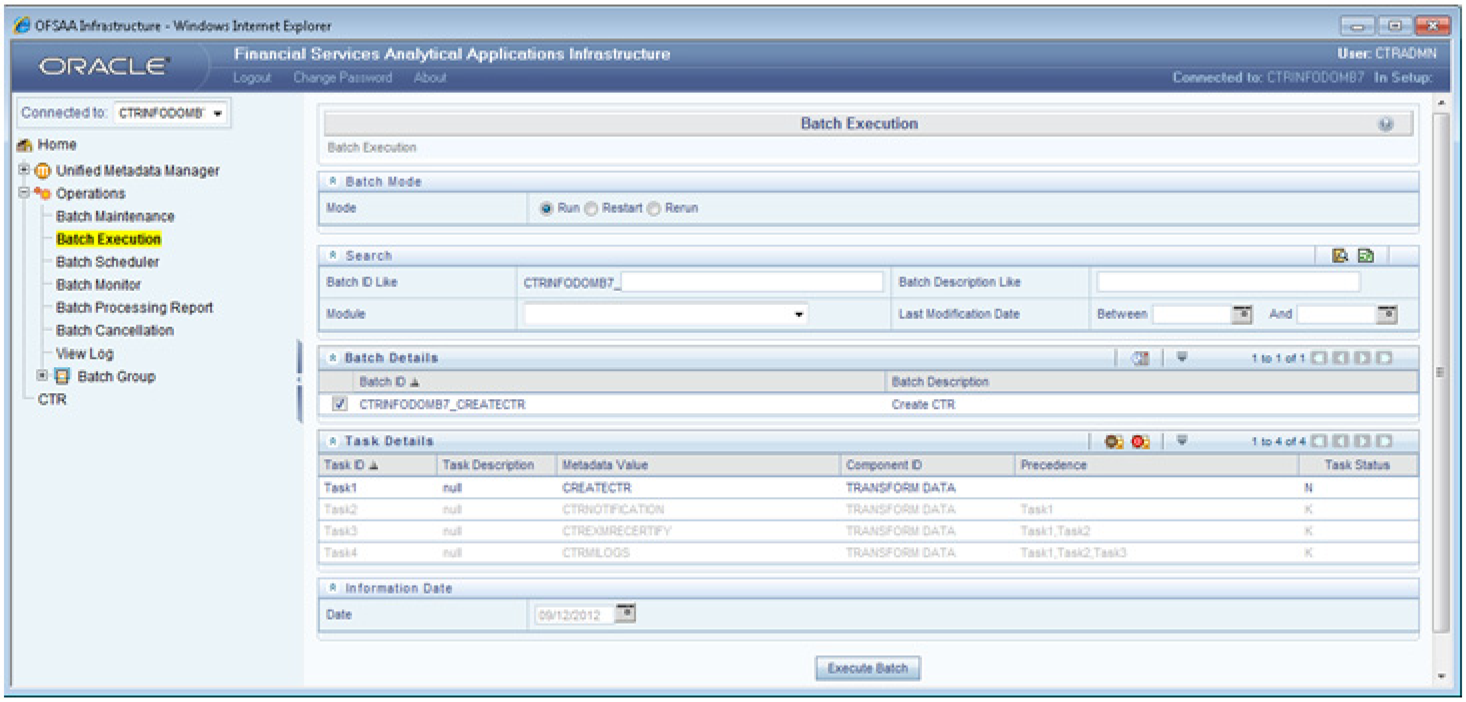
Task ID	Task Description	Metadata Value	Component ID	Precedence	Task Status
Task1	null	CREATECTR	TRANSFORM DATA		N
Task2	null	CTRNOTIFICATION	TRANSFORM DATA	Task1	K
Task3	null	CTREXMRECERTIFY	TRANSFORM DATA	Task1,Task2	K
Task4	null	CTRMLOGS	TRANSFORM DATA	Task1,Task2,Task3	K

Information Date

Date:

Figure 16. CTR Batch Execution - Highlighted Tasks Details

9. Select an information date. Click on the calendar icon, and choose the processing date as information date



Batch Execution

Batch Mode

Mode: [● Run] ○ Restart ○ Rerun

Search

Batch ID Like	CTRINFODOMB7_	Batch Description Like	
Module		Last Modification Date	Between And

Batch Details — 1 to 1 of 1

	Batch ID	Batch Description
✓	CTRINFODOMB7_CREATECTR	Create CTR

Task Details — 1 to 4 of 4

Task ID	Task Description	Metadata Value	Component ID	Precedence	Task Status
Task1	null	CREATECTR	TRANSFORM DATA		N
Task2	null	CTRNOTIFICATION	TRANSFORM DATA	Task1	K
Task3	null	CTREXMRECERTIFY	TRANSFORM DATA	Task1,Task2	K
Task4	null	CTRMLOGS	TRANSFORM DATA	Task1,Task2,Task3	K

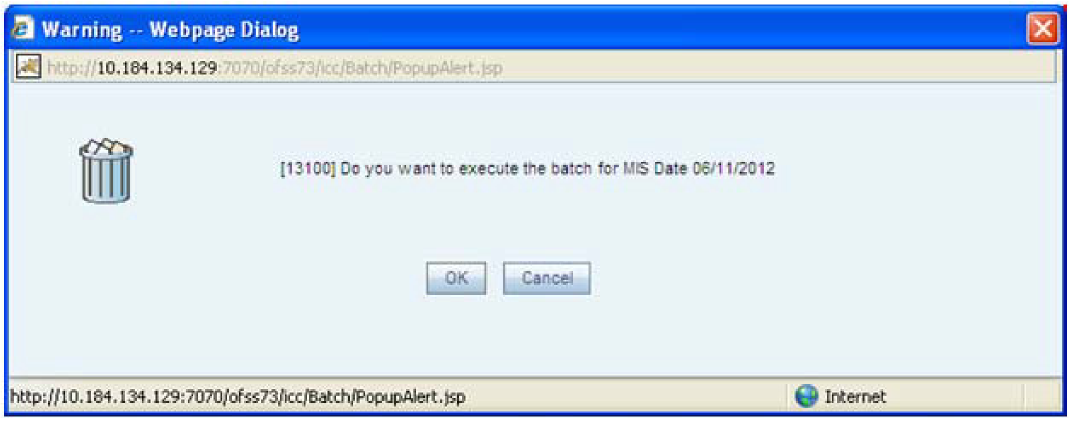
Information Date

Date: 09/12/2012

Figure 17. CTR Batch Execution - Select Information Date

10. Click **Execute** to run the batch for the provided processing date.

    The following message is displayed:

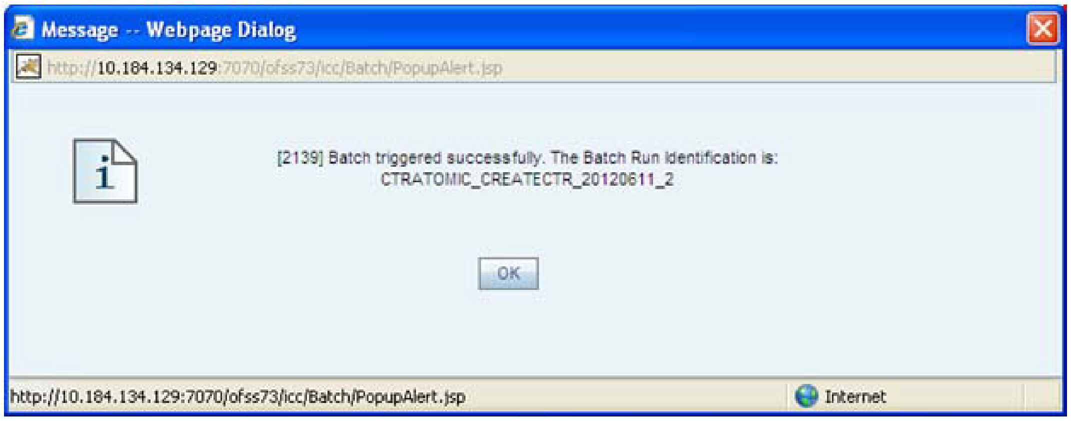

Warning -- Webpage Dialog
http://10.184.134.129:7070/ofss73/icc/Batch/PopupAlert.jsp
OK
Cancel
http://10.184.134.129:7070/ofss73/icc/Batch/PopupAlert.jsp
Internet

[13100] Do you want to execute the batch for MIS Date 06/11/2012

Figure 18. CTR Batch Execution Warning Window

11. Click **OK**.

A pop-up window confirming the successful execution of the batch is displayed.


Message -- Webpage Dialog
http://10.184.134.129:7070/ofss73/icc/Batch/PopupAlert.jsp
OK
http://10.184.134.129:7070/ofss73/icc/Batch/PopupAlert.jsp
Internet

[2139] Batch triggered successfully. The Batch Run Identification is: CTRATOMIC_CREATECTR_20120611_2

Figure 19. CTR Batch Execution Confirmation Message Window

12. Click **OK**.

About CTR Web Service Invocation

In order to invoke the CTR web service, you must use the following details:

- End Point : `<<CTRContext /services/ExcemptionCheck`
  - **CTRContext** Format is "`<<protocol>>://<<host>>:<<port>>/<<DeploymentName>>`" e.g. `http://localhost:8050/CTR`
  - wsdl can be taken from the same end point to generate client-stub
- UserName: ctruser
- Password: For information about updating the Exemption Password, refer to *Configuring the Exemption Password*, on page 109.

CHAPTER 6

Batch Processing Utilities

Oracle Financial Services Behavior Detection Platform provides utilities that enable you to set up and modify a selection of batch-related database processes. The chapter focuses on the following topics:

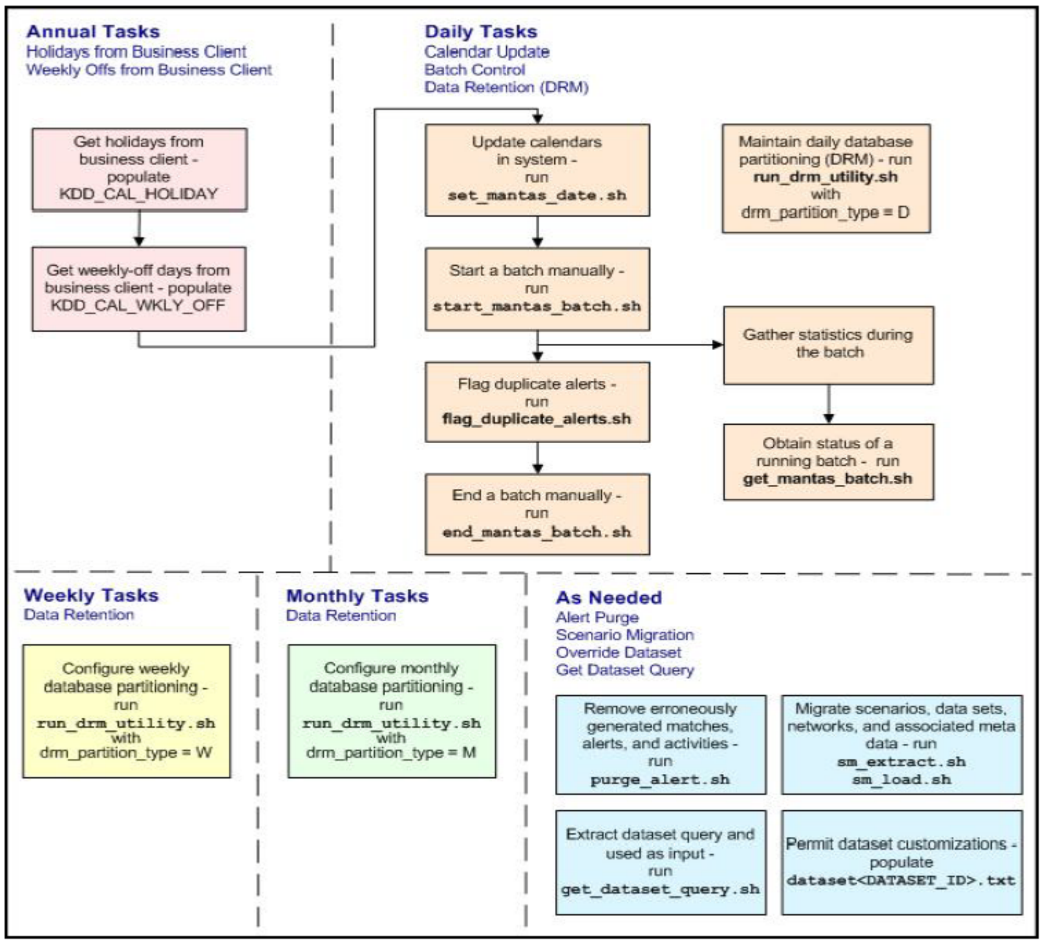
- About Administrative Utilities
- About Annual Activities
- Alert Purge Utility
- Batch Control Utility
- Calendar Manager Utility
- Data Retention Manager
- Database Statistics Management

About Administrative Utilities

Behavior Detection database utilities enable you to configure and perform batch-related system pre-processing and post-processing activities.

- **Alert Purge:** Provides the capability to remove erroneously generated matches, alerts, and activities (See *Alert Purge Utility*, on page 79 for more information).
- **Batch Control:** Manages the start and termination of a batch process (from Data Ingestion to alert post-processing) and enables access to the currently running batch (See *Batch Control Utility*, on page 86 for more information).
- **Calendar Manager**: Updates calendars in the Oracle Financial Services Behavior Detection Platform system based on predefined business days, holidays, and days off, or non-business days (See *Calendar Manager Utility*, on page 93 for more information).
- **Data Retention Manager:** Provides the capability to manage the processing of partitioned tables in Behavior Detection. This utility purges data from the system based on configurable retention period defined in database (See *Data Retention Manager*, on page 97 for more information).

Figure 15 illustrates the frequency with which you use these batch-related database utilities when managing activities: daily, weekly, monthly, annually, or as needed


Annual Tasks
Holidays from Business Client
Weekly Offs from Business Client
Get holidays from business client - populate KDD_CAL_HOLIDAY
Get weekly-off days from business client - populate KDD_CAL_WKLY_OFF
Daily Tasks
Calendar Update
Batch Control
Data Retention (DRM)
Update calendars in system - run set_mantas_date.sh
Start a batch manually - run start_mantas_batch.sh
Flag duplicate alerts - run flag_duplicate_alerts.sh
End a batch manually - run end_mantas_batch.sh
Maintain daily database partitioning (DRM) - run run_drm_utility.sh with drm_partition_type = D
Gather statistics during the batch
Obtain status of a running batch - run get_mantas_batch.sh
Weekly Tasks
Data Retention
Configure weekly database partitioning - run run_drm_utility.sh with drm_partition_type = W
Monthly Tasks
Data Retention
Configure monthly database partitioning - run run_drm_utility.sh with drm_partition_type = M
As Needed
Alert Purge
Scenario Migration
Override Dataset
Get Dataset Query
Remove erroneously generated matches, alerts, and activities - run purge_alert.sh
Migrate scenarios, data sets, networks, and associated meta data - run sm_extract.sh sm_load.sh
Extract dataset query and used as input - run get_dataset_query.sh
Permit dataset customizations - populate dataset<DATASET_ID>.txt

.

Figure 20. Managing Database Activities with Utilities

Common Resources for Administrative Utilities

Configuration files enable the utilities to share common resources such as database configuration, directing output files, and setting up logging activities. Common resources include the following:

- `install.cfg file`
- `categories.cfg File`

`install.cfg` file

Configuration information resides in the `<INSTALL_DIR>/database/db_tools/mantas_cfg/install.cfg` configuration file. The configuration file contains modifiable instructions for Oracle database drivers and provides information that each utility requires. It also provides the user name and password that you need to connect to the database. In this file, you can modify values of specific utility parameters, change the locations of output files, and specify database details for extraction and data loading.

The `install.cfg file` contains information unique to each utility and common configuration parameters; headings in the file clearly identify a utility's parameters. You can also modify the current logging configuration (for example, activate or deactivate particular logging levels and specify locations for logging entries).

Figure 21 provides a sample install.cfg file with common and utility-specific information. Logging information appears at the end of the file.

Note: You should ensure that all schema names (that is, `MANTAS, BUSINESS, and MARKET`) are in uppercase.

```
# This configuration file supports the following database utilities:
#  Calendar Mangager
#  Batch Control
#  Truncate Manager
#  Scenario Migration
#  Alert Purge
#  Data Retention Manager
#  Email Notification
#  Data Analysis Tool
#
# The file contains some properties that are common and specific properties for each
# of the tools.

################ COMMON CONFIGURATION ENTRIES #######################


database.driverName=oracle.jdbc.driver.OracleDriver
utils.database.urlName=jdbc:oracle:oci:@Ti5O10S10
utils.database.username=DB_UTIL_USER_TEST58
utils.database.password=DB_UTIL_USER_TEST58


schema.mantas.owner=mantas_TEST58
utils.miner.user=KDD_MNR_TEST58
utils.miner.password=
utils.altio.username=KDD_ALTIO_TEST58
schema.business.owner=BUSINESS_TEST58
schema.market.owner=MARKET_TEST58
utils.data.directory=/users/mantast/Solaris10_mantas58_b09_Ti5O10S10_Iron_13080_WAS/databa
se/db_tools/data
ingest.user=INGEST_USER_TEST58
ingest.password=


################ CALENDAR MANAGER CONFIGURATION ##################
# The look back and look forward days of the provided date.
# These values are required to update the KDD_CAL table. The maximum look back or forward
# is 999 days.
calendar.lookBack=400
calendar.lookForward=14
############### BATCH CONTROL CONFIGURATION #####################
# When ending the batch, age alerts in calendar or business days
age.alerts.useBusinessDays=Y
```

(*Continued on next page*)

(*Continued from previous page*)

```
############### TRUNCATE MANAGER ################################
# Specify the database username and password for truncation manager
truncate.database.username=${ingest.user}
truncate.database.password=${ingest.password}

################ SCENARIO MIGRATION CONFIGURATION #######################

#### GENERAL SCENARIO MIGRATION SETTINGS

#Specify the flags for whether scoring rules and wrapper datasets need to be extracted or
loaded
score.include=N
wrapper.include=N

#Specify the Use Code for the scenario. Possible values are 'BRK' or 'EXP'
load.scnro.use=BRK

#Specify the full path of depfile and name of fixfile used for extraction and loading
#Note : fixfile need not be specified in case of loading
sm.depfile=/users/mantast/Solaris10_mantas58_b09_Ti5O10S10_Iron_13080_WAS/database/db_tool
s/mantas_cfg/dep.cfg

sm.release=1.1

#### EXTRACT
# Specify the database details for extraction
extract.database.username=${utils.database.username}
extract.database.password=${utils.database.password}

# Specify the jdbc driver details for connecting to the source database
extract.conn.driver=${database.driverName}
extract.conn.url=jdbc:oracle:oci:@Ti5O10S10

#Source System Id
extract.system.ID=

# Specify the schema names for Extract
extract.schema.mantas=${schema.mantas.owner}
extract.schema.business=${schema.business.owner}
```

(*Continued on next page*)

(Continued from previous page)

```
extract.schema.market=${schema.market.owner}
extract.user.miner=${load.user.miner}
extract.miner.password=${utils.miner.password}

# File Paths for Extract

#Specify the full path in which to place extracted scenarios
extract.dirname=/users/mantast/Solaris10_mantas58_b09_Ti5O10S10_Iron_13080_WAS/database/db
_tools/data

#Specify the full path of the directory where the backups for the extracted scripts would be
maintained
extract.backup.dir=/users/mantast/Solaris10_mantas58_b09_Ti5O10S10_Iron_13080_WAS/database
/db_tools/data/temp

#Controls whether jobs and thresholds are constrained to IDs in the product range
(product.ID.range.min
# through product.ID.range.max). Values are Y and N. If the range is not restriced, you can
use range.check
# to fail the extract if there are values outside the product range.
extract.product.range.only=N
extract.product.range.check=N

#### LOAD

# Specify the jdbc driver details for connecting to the target database
load.conn.driver=${database.driverName}
load.conn.url=${utils.database.urlName}

#Target System ID
load.system.ID=Ti5O10S10

# Specify the schema names for Load
load.schema.mantas=${schema.mantas.owner}
load.schema.business=${schema.business.owner}
load.schema.market=${schema.market.owner}
load.user.miner=${utils.miner.user}
load.miner.password=${utils.miner.password}

#Directory where scenario migration files reside for loading
load.dirname=/users/mantast/Solaris10_mantas58_b09_Ti5O10S10_Iron_13080_WAS/database/
db_tools/data

# Specify whether threshold can be updated
load.threshold.update=Y

# Specify whether or not to verify the target environment on load
verify.target.system=N

################ ALERT PURGE CONFIGURATION ##########################
# Set the Alert Purge input variables here.
# (use the word "null" as the value of any parameters that are not
#  to be used)
```

(Continued on next page)

(Continued from previous page)

```
limit_matches=N
purge=Y
batch_size=5000
job=null
scenario=null
# enter dates, with quotes in the following format:
#   'DD-MON-YYYY HH24:MI:SS'
start_date=null
end_date=null
alert_status=NW

#Base Working Directory required to put the temporary log from Database Server
ap.storedproc.logdir=/tmp

#The common Path required to put the SQL files to execute
commonSQLFilePath=/users/mantast/Solaris10_mantas58_b09_Ti5O10S10_Iron_13080_WAS/database/
db_tools/data

######### DATA RETENTION MANAGER CONFIGURATION ########################
#
# Set the Data Retention Manager input variables here.
##
drm_operation=P
drm_partition_type=D
drm_owner=${schema.business.owner}
drm_object_name=A
drm_weekly_proc_fl=N

######### Email Notification ##########################################
#
# The following sections contain information on configuring email
# notification information. If you wish to use Exchange, you must purchase
# Java Exchange Connector, obtain a license and the jec.jar file. The license
# file must be placed in the mantas_cfg file, and the jec.jar file must be
# copied to the db_tools/lib directory. Then, edit the file
# db_tools/bin/run_push_email.ksh, uncomment the JEC_JARS= line.
#
#######################################################################
# Currently only smtp, smtps, or exchange
email.type=smtp

# Number of notifications that can run in parallel
notification.threads=4

# Max number of active db connections
utils.database.max_connections=4

# From address for sent mails. This is ignored in Exchange mode. If omitted in SMTP mode,
the mail account associated

# with the Unix/Linux account is used.

email.from=
```

(Continued on next page)

(Continued from previous page)

```
# SMTP settings
email.smtp.host=

# smtp port is usually 25 for smtp, 465 for smtps
email.smtp.port=25
email.smtp.auth=false
email.smtp.user=
email.smtp.password=
email.smtp.useHTML=true

# Exchange settings *** See above for instructions to enable this ***
#  Your Exchange administrator should help identify these settings
#
email.exchange.server=
email.exchange.domain=
email.exchange.user=
email.exchange.password=
email.exchange.prefix=Exchange
email.exchange.mailbox=
email.exchange.useSSL=true
email.exchange.useFBA=true
email.exchange.useNTLM=false
email.exchange.draftsfoldername=drafts
email.exchange.useHTML=true

#HTML email styles
email.style.header=font-family:Arial, Helvetica, sans-serif;font-size:10pt; color:black;
email.style.hr=color: #555; background-color: #f00; height: 1px;
email.style.title=font-family:Arial, Helvetica, sans-serif;font-style:
bold;font-size:12pt;
email.style.message=font-family:Arial, Helvetica, sans-serif;font-size:11pt;
email.style.table=font-family:Arial, Helvetica, sans-serif;border:1px solid #000;
border-collapse:collapse;

email.style.th=font-style: bold;border:1px solid #000; border-collapse:collapse; padding:
4px; background:#C7DAED
email.style.tr=font-size:10pt
email.style.td=border:1px solid #000; border-collapse:collapse; padding: 4px
email.style.footer=font-family:Arial, Helvetica, sans-serif;font-size:10pt; color:black;
```

(Continued on next page)

(Continued from previous page)

```
email.style.disclaimer=font-style: italic;
######### HIGHLIGHTS GENERATION CONFIGURATION #########################
# Set the default currency code.
# See /mantas_cfg/etc/xml/CUR_Currencies.xml for supported currency
# codes.
#
currency.default=USD

######### HDC CONFIGURATION #########################
# Set the maximum number of hdc threads.
#
hdc.maxthreads=1
hdc.batchsize=10000

######### Data Analysis Tool CONFIGURATION #########################
# Username and password for connecting to the database

dat.database.username=${ingest.user}
dat.database.password=${ingest.password}

# Input file for analysis
dat.analysis.input=/users/mantast/Solaris10_mantas58_b09_Ti5O10S10_Iron_13080_WAS/database
/db_tools/mantas_cfg/analysis_aml.xml

# Output file and file format control
dat.analysis.output=/users/mantast/Solaris10_mantas58_b09_Ti5O10S10_Iron_13080_WAS/databas
e/db_tools/data/analysis.html

# Valid values for dat.output.format are HTML and TEXT
dat.output.format=HTML

# Delimiter only applies to TEXT output format
dat.output.delimiter=,
log.fatal.synchronous=false
log.warning.synchronous=false
log.notice.synchronous=false
log.diagnostic.synchronous=false
log.trace.synchronous=true
```

(Continued on next page)

(Continued from previous page)

```
# Specify the hostname of syslog if syslog was chosen as the log output location
# anywhere above.
# Logging will go to the console if syslog was selected and this property is
# not given a value.
log.syslog.hostname=

# Specify the hostname of the SMTP server if an e-mail address was chosen as
# the log output location anywhere above.
# Logging will go to the console if an e-mail address was selected and this
# property is not given a value.
log.smtp.hostname=

# Specify the maxfile size of a logfile before the log messages get rolled to
# a new file (measured in MBs).
# If this property is not specified, the default of 10 MB will be used.
log.max.size=

#NOTE: The values for the following variables need not be changed
# Specify the ID range for wrapper datasets
dataset.wrapper.range.min=113000001
dataset.wrapper.range.max=114000000
product.ID.range.min=113000000
product.ID.range.max=200000000
```

Figure 21. Sample install.cfg File

`categories.cfg` File

In the `<INSTALL_DIR>/database/db_tools/mantas_cfg/categories.cfg` file, you can modify the default location to where you want to direct logging output for each utility. The entries that you make require a specific format; the file contains instructions and examples of correct formatting. Figure provides a sample `categories.cfg` file.

```
# Common Logging categories configuration for Oracle Financial Services Database
#
# Specify the log location for messages of a specific category.
# The property name should be of the form: log.category.{CATEGORY_NAME}.location
# If logging to a category that is not specified below, the messages are logged to
# a configurable default location.
# Valid values are console, syslog, eventviewer, mantaslog, an e-mail address, or the
# full path to a file.
# If specifying mantaslog, also specify the property log.mantaslog.location with
# the desired path to the logfile in install.cfg. If running the algorithms, use the
# format job<job #>-datetimestamp for the mantaslog filename. For other subsystems, the
# format is mantaslog-datetimestamp.
#
# NOTE: Category names cannot contain the following reserved words: fatal,
# warning, notice, diagnostic, trace, category, or location.
# List multiple locations for each property by using a comma delimiter.
#
# NOTE: These are commented out because Oracle Financial Services does not currently route
# category. Entries are placed in the configured default location in install.cfg.
# These can be uncommented and modified if routing by category is necessary.
#
log.category.ALERT_PURGE.location=/users/orion/mantas1.1/database/db_tools/logs/
alert_purge.log
log.category.BATCH_CONTROL.location=/users/orion/mantas1.1/database/db_tools/logs/
batch_control.log
log.category.CALENDAR_MANAGER.location=/users/orion/mantas1.1/database/db_tools/logs/
calendar_manager.log
log.category.DATA_RETENTION_MANAGER.location=/users/orion/mantas1.1/database/db_tools/
logs/DRM_Utility.log
log.category.TRUNCATE_MANAGER.location=/users/orion/mantas1.1/database/db_tools/logs/
truncate_manager.log
log.category.COMMON_UTILITIES.location=/users/orion/mantas1.1/database/db_tools/logs/
common_utilities.log
log.category.EXTRACT.location=/users/orion/mantas1.1/database/db_tools/logs/extract.log
log.category.LOAD.location=/users/orion/mantas1.1/database/db_tools/logs/load.log
```

(Continued on next page)

(Continued from previous page)

```
log.category.REFRESH_TEMP_TABLE.location=/users/orion/mantas1.1/database/db_tools/logs/
refresh_temp_table.log
log.category.RUN_STORED_PROCEDURE.location=/users/orion/mantas1.1/database/db_tools/logs/
run_stored_procedure.log
log.category.GET_DATASET_QUERY.location=/users/orion/mantas1.1/database/db_tools/logs/
get_dataset_query.log
log.category.PUSH_EMAIL.location=/users/orion/mantas1.1/database/db_tools/logs/
push_email.log
log.category.HIGHLIGHT_GENERATOR.location=/users/orion/mantas1.1/database/db_tools/logs/
highlight_generator.log
log.category.REPORT.location=/users/orion/mantas1.1/database/db_tools/logs/report.log
log.category.DATA_ANALYSIS_TOOL.location=/users/orion/mantas1.1/database/db_tools/logs/
data_analysis_tool.log


# Specify the location of messages of a specific severity and category.
# The valid values are the same as for category.
# List multiple locations for each property by using a comma delimiter.
# If an entry for a severity does not appear here, the message is logged to
# the location specified for the category by the above property. If that
# does not exist, it is logged to the configured default location in install.cfg.
#
# NOTE: The entry below is just an example. It is commented out because mantas
# does not route by category/severity. These can be uncommented and modified if
# routing by category/severity is necessary.
#
#log.EXAMPLE_CATEGORY.warning.location=syslog
```

Figure 22. Sample Logging Information in the categories.cfg File

Configuring Console Output

Figure 23 displays a section of the sample `categories.cfg` file from Figure . Note the log routing information in bold text.

```
log.category.ALERT_PURGE.location=console,/users/orion/mantas1.1/database/db_tools/logs/
  alert_purge.log
log.category.BATCH_CONTROL.location=/users/orion/mantas1.1/database/db_tools/logs/
  batch_control.log
log.category.CALENDAR_MANAGER.location=console,/users/orion/mantas1.1/database/db_tools/
  logs/calendar_manager.log
log.category.DATA_RETENTION_MANAGER.location=/users/orion/mantas1.1/database/db_tools/
  logs/DRM_Utility.log
log.category.TRUNCATE_MANAGER.location=/users/orion/mantas1.1/database/db_tools/logs/
  truncate_manager.log
log.category.COMMON_UTILITIES.location=/users/orion/mantas1.1/database/db_tools/logs/
  common_utilities.log
log.category.EXTRACT.location=/users/orion/mantas1.1/database/db_tools/logs/extract.log
log.category.LOAD.location=/users/orion/mantas1.1/database/db_tools/logs/load.log
log.category.REFRESH_TEMP_TABLE.location=/users/orion/mantas1.1/database/db_tools/logs/
  refresh_temp_table.log
log.category.RUN_STORED_PROCEDURE.location=/users/orion/mantas1.1/database/db_tools/logs/
  run_stored_procedure.log
log.category.GET_DATASET_QUERY.location=/users/orion/mantas1.1/database/db_tools/logs/
  get_dataset_query.log
log.category.PUSH_EMAIL.location=/users/orion/mantas1.1/database/db_tools/logs/
  push_email.log
log.category.HIGHLIGHT_GENERATOR.location=/users/orion/mantas1.1/database/db_tools/logs/
  highlight_generator.log
log.category.REPORT.location=/users/orion/mantas1.1/database/db_tools/logs/report.log
log.category.DATA_ANALYSIS_TOOL.location=/users/orion/mantas1.1/database/db_tools/logs/
  data_analysis_tool.log
```

Figure 23. Sample Log Routing Information

The bolded text in the above example (**`console,`**) implies that a specific utility displays logging information at the console in addition to recording the information in the appropriate log file. In Figure 23, Alert Purge and Calendar Manager display relevant utility information in addition to logging it. If an entry in the `categories.cfg` file does not already include this information, you must add it manually, including the comma.

About Annual Activities

Behavior Detection requires that you perform certain calendar management tasks at least annually: loading holidays and weekly off-days from an Oracle client. This ensures that the system has the necessary information for populating its own business calendars.

This section covers the following topics:

- Loading Holidays
- Loading Non-business Days

Loading Holidays

Typically, on an annual basis, you populate holidays for the upcoming calendar year into the Behavior Detection `KDD_CAL_HOLIDAY` database table. This ensures that the table contains holidays for at least the next year. Figure 24 provides an example of a SQL script for loading the table.

```
INSERT INTO KDD_CAL_HOLIDAY ( CLNDR_NM, CLNDR_DT, HLDY_NM,
HLDY_TYPE_CD ) VALUES ( 'SYSCAL', TO_DATE( '01/01/2006',
'MM/DD/YYYY'), 'New Year''s Day - 2006', 'C');


INSERT INTO KDD_CAL_HOLIDAY ( CLNDR_NM, CLNDR_DT, HLDY_NM,
HLDY_TYPE_CD ) VALUES ( 'SYSCAL', TO_DATE( '01/16/2006',
'MM/DD/YYYY'), 'Martin Luther King Jr.''s Birthday - 2006', 'C');


INSERT INTO KDD_CAL_HOLIDAY ( CLNDR_NM, CLNDR_DT, HLDY_NM,
HLDY_TYPE_CD ) VALUES ( 'SYSCAL', TO_DATE( '02/20/2006',
'MM/DD/YYYY'), 'President''s Day - 2006', 'C');


INSERT INTO KDD_CAL_HOLIDAY ( CLNDR_NM, CLNDR_DT, HLDY_NM,
HLDY_TYPE_CD ) VALUES ( 'SYSCAL', TO_DATE( '04/14/2006',
'MM/DD/YYYY'), 'Good Friday - 2006', 'C');


INSERT INTO KDD_CAL_HOLIDAY ( CLNDR_NM, CLNDR_DT, HLDY_NM,
HLDY_TYPE_CD ) VALUES ( 'SYSCAL', TO_DATE( '05/29/2006',
'MM/DD/YYYY'), 'Memorial Day - 2006', 'C');


INSERT INTO KDD_CAL_HOLIDAY ( CLNDR_NM, CLNDR_DT, HLDY_NM,
HLDY_TYPE_CD ) VALUES ( 'SYSCAL', TO_DATE( '07/04/2006',
'MM/DD/YYYY'), 'Independence Day - 2006', 'C');


INSERT INTO KDD_CAL_HOLIDAY ( CLNDR_NM, CLNDR_DT, HLDY_NM,
HLDY_TYPE_CD ) VALUES ( 'SYSCAL', TO_DATE( '09/04/2006',
'MM/DD/YYYY'), 'Labor Day - 2006', 'C');


INSERT INTO KDD_CAL_HOLIDAY ( CLNDR_NM, CLNDR_DT, HLDY_NM,
HLDY_TYPE_CD ) VALUES ( 'SYSCAL', TO_DATE( '11/22/2006',
'MM/DD/YYYY'), 'Thanksgiving Day - 2006', 'C');


INSERT INTO KDD_CAL_HOLIDAY ( CLNDR_NM, CLNDR_DT, HLDY_NM,
HLDY_TYPE_CD ) VALUES ( 'SYSCAL', TO_DATE( '12/25/2006',
'MM/DD/YYYY'), 'Christmas Day - 2006', 'C');


COMMIT;
```

Figure 24. Sample KDD_CAL_HOLIDAY Table Loading Script

Table 23 provides the contents of the `KDD_CAL_HOLIDAY` table.

Table 23. KDD_CAL_HOLIDAY Table Contents

Column Name	Description
`CLNDR_NM`	Specific calendar name.
`CLNDR_DT`	Date that is a holiday.
`HLDY_NM`	Holiday name (for example, Thanksgiving or Christmas).
`HLDY_TYPE_CD`	Indicates whether the business is Closed (C) or Shortened (S).
`SESSN_OPN_TM`	Indicates the opening time of the trading session for a shortened day. The format is HHMM.
`SESSN_CLS_TM`	Indicates the closing time of the trading session for a shortened day. The format is HHMM.
`SESSN_TM_OFFSET_TX`	Indicates the timezone offset for SESSN_OPN_TM and SESSN_CLS_TM.

When the system runs the `set_mantas_date.sh` script, it queries the `KDD_CAL_HOLIDAY` table for the maximum date for each calendar in the table. If the maximum date is less than 90 days ahead of the provided date, the process logs a warning message that the specific calendar's future holidays need updating. If any calendars have no holiday records, the system logs a Warning message that the specific calendar has no recorded holidays for the appropriate date range.

Loading Non-business Days

After obtaining non-business days (or *weekly off-days*; typically Saturday and Sunday) from an Oracle client, load this information for the upcoming calendar year into the `KDD_CAL_WKLY_OFF` table.

Figure 25 provides an example of a SQL script for loading the table.

```
INSERT INTO KDD_CAL_WKLY_OFFS (CLNDR_NM, DAY_OF_WK) VALUES (
 'SYSCAL', 1);

INSERT INTO KDD_CAL_WKLY_OFFS (CLNDR_NM, DAY_OF_WK) VALUES (
 'SYSCAL', 7);

COMMIT;
```

Figure 25. Sample KDD_CAL_HOLIDAY Table Loading Script

Note: By default, the system identifies Saturdays and Sundays as non-business days in the system calendar (SYSCAL).

Table 24 provides the contents of the `KDD_CAL_WKLY_OFF` table.

Table 24. KDD_CAL_WKLY_OFF Table Contents

Column Name	Description
`CLNDR_NM`	Specific calendar name.
`DAY_OF_WK`	Value that represents the day of the week: Sunday=1, Monday=2, Tuesday=3 ... Saturday=7.

If the table does not contain records for any calendar in the list, the system logs a Warning message that the specific calendar contains no weekly off-days.

Alert Purge Utility

Occasionally, ingestion of certain data results in the creation of false matches, alerts, and activities. While correction and data re-ingestion is possible, the system does not remove these erroneously generated matches, alerts, and activities automatically.

The Alert Purge Utility enables you to identify and remove such matches, alerts, and activities selectively, based on the Behavior Detection Job ID or Behavior Detection Scenario ID and a date range with optional alert status codes. Additional parameters enable you to simulate a purge run to determine all found matches, alerts, and activities using the input parameters. You can also limit the alerts in the purge process only to those that contain false matches.

The utility consists of a UNIX shell script, Java executables, and a configuration file in which you define the process parameters to use in the purge processing. The system directs output to a configurable log file; processing appends this log with information about subsequent executions of the scripts.

This section covers the following topics:

- Directory Structure
- Logs
- Precautions
- Using the Alert Purge Utility
- Sample Alert Purge Processes

Directory Structure

Table 25 provides the directory structure for the Alert Purge Utility.

Table 25. Alert Purge Utility Directory Structure

Directory	Description
bin/	Contains executable files, including the `run_alert_purge.sh` shell script.
lib/	Contains required class files in .jar format.
mantas_cfg/	Contains configuration files (for example, `install.cfg` and `categories.cfg`), in which you can configure properties and logging attributes.
logs/	Keeps the <INSTALL_DIR>/database/db_tools/logs/ Alert_Purge.log file that the utility generates during execution.
data/	Keeps .sql files for execution.

Logs

As the Alert Purge Utility performs alert detection activities, it generates a log that it enters in the `<INSTALL_DIR>/database/db_tools/logs/Alert_Purge.log` file (the logging process time-stamps all entries). The log file contains relevant information such as status of the purge processing, log-relevant information, and error records.

You can modify the current logging configuration for the Alert Purge Utility in the `<INSTALL_DIR>/database/db_tools/mantas_cfg/install.cfg` and `categories.cfg` files. For more information about logging in these configuration files, See *Common Resources for Administrative Utilities,* on page 66, and Appendix A, *Logging*, on page 133 for more information.

Precautions

You use the utility to rid the system of falsely-generated matches and alerts. Other than recorded information in the `<INSTALL_DIR>/database/db_tools/logs/`

`Alert_Purge.log` file, the system does not capture audit information for this process. The utility does not update other alerts' prior counts as a result of purging alerts.

Note: You cannot purge an alert that is used to trigger Auto Suppression. You can tell if an alert ID is used to trigger Auto Suppression by looking at the `kdd_auto_suppr_alert.trgr_alert_id` column to see if it contains the alert ID in question. If so, you have to delete the record before attempting to purge the alert.

Run the Alert Purge Utility:

- Through one process at a time. Multiple, simultaneous executions of the utility may lead to unexpected results and compromise the relational integrity of match, alert, and action data.
- When no users are editing or viewing any of the alerts, actions, or associated information (including matches derived from the alerts and actions specified, alerts derived from the specified actions, and actions derived from the specified alerts). However, you can run the utility during editing or viewing of other alerts and related information. You can also run the utility during alert post-processing, subject to time constraints.

Using the Alert Purge Utility

The Alert Purge Utility is not part of an automated batch process that an application such as Maestro or Unicenter AutoSys controls. You run this manual process only when necessary . The following sections describe configuring and executing the utility, as well as the utility's process flow:

- Configuring the Alert Purge Utility
- Executing the Alert Purge Utility
- Processing for Purging

Configuring the Alert Purge Utility

The `<INSTALL_DIR>/database/db_tools/mantas_cfg/install.cfg file` contains common configuration information that the Alert Purge Utility and other utilities require for processing (See Figure 26). The following sample section from the `install.cfg` file provides configuration information specific to this utility.

```
################ ALERT PURGE CONFIGURATION ########################
# Set the Alert Purge input variables here..
# (set the job/scenario value you DO NOT USE to null)
#

limit_matches=Y
purge=N
batch_size=5000
job=null
scenario=null
# Enter dates with quotes in the following format:
# 'DD-MMM-YYYY HH:MI:SS' or 'DD-MON-YYYY'.
start_date=null
end_date=null
alert_status=NW
#Base Working Directory required to put the temporary log from the
#Database server.
ap.storedproc.logdir=/tmp
```

Figure 26. Configuration Information

Note: Not specifying a value of null (for example, leaving a value blank) in this section of the `install.cfg` file causes undesirable results.

Table 26 describes required and optional parameters for this utility.

Table 26. Alert Purge Utility Parameters

Parameter	Description
`purge`	Determines how the utility performs processing, depending on the specified value: ● N (default): Performs all processing up to the point of the purge. The utility identifies resulting matches, alerts, and actions, but performs no purging. ● Y: Performs the above in addition to purging matches, alerts, and actions.
`limit_matches`	Identifies restrictions on the matches to delete: ● Y (default): If a match that you want to delete is part of an alert that contains matches that you do not want to delete, do not delete this match either (applies to multi-match alerts). ● N: Deletes all selected matches for purging based on the input criteria. The utility deletes only alerts and associated actions that exclusively contain matches to be purged. **Note:** The system purges matches that do not relate to alerts, regardless of the value of limit_matches.
`batch_size`	*Optional:* Sets the batch size of purge actions to minimize log space use. Specifying a non-positive value or specifying no value uses the default of 5,000 rows.
`job`	Identifies the Behavior Detection Job ID to purge (value in the `JOB_ID` column of the `KDD_JOB t`able). Selecting this variable causes the system to ignore the scenario, `start_date, end_date, and alert_status` variables. **Note:** If you assign a value to the job parameter, do not assign a value to the scenario parameter. Likewise, if you assign a value to scenario, assign a value of NULL to job. If both the Job ID and the Scenario ID are assigned values, the Alert Purge Utility continues to run using the Job ID, ignoring the Scenario ID.
`scenario`	Identifies the Behavior Detection scenario ID to purge (value in the `SCNRO_ID` column of the `KDD_SCNRO` table). **Note:** If you assign a value to scenario, assign a value of NULL to job. Likewise, if you assign a value to job, assign a value of NULL to scenario. If both the Job ID and the Scenario ID are assigned values, the Alert Purge Utility continues to run using the Job ID, ignoring the Scenario ID.
`start_date`	Indicates the start date for the Scenario ID (when the scenario parameter is in use), in the format `'DD-MON-YYYY HH:MM:SS' or 'DD-MON-YYYY'.` When using only the date, the time component defaults to midnight. You must set this parameter to NULL if it is not used. However, when using the scenario parameter, it cannot be set to NULL.

Table 26. Alert Purge Utility Parameters

Parameter	Description
`end_date`	Indicates the end date for the Scenario ID (when the scenario parameter is in use), in the format 'D`D-MON-YYYY HH:MM:SS' or.'DD-MON-YYYY'` When using only the date, the time component defaults to midnight. You must set this parameter to NULL if it is not used. However, when using the scenario parameter, it cannot be set to NULL.
`alert_status`	Identifies an alert status code (when the scenario parameter is in use) against which to restrict the Alert Purge Utility further. (Comma-separated list.) Alert status codes include: NW (New), OP (Open), CL (Closed), FL, RO and RA. When using the scenario parameter, the `alert_status` must be used, however, you can set it to NULL.

Executing the Alert Purge Utility

To execute the Alert Purge Utility, follow these steps:

1. Verify that the Behavior Detection database is operational: `tnsping <database instance name>`
2. Verify that the `<INSTALL_DIR>/database/db_tools/mantas_cfg/ install.cfg` configuration file contains the correct source database connection and logging information.
3. Access the directory where the shell script resides:
   `cd <INSTALL_DIR>/database/db_tools/bin`
4. Start the alert purge shell script:
   `run_alert_purge.sh`

   Executing this command sets the environment classpath and starts the utility.

Processing for Purging

Upon execution of the `run_alert_purge.sh` script, the Alert Purge Utility generates a listing of actions, matches, and alerts that it needs to purge, and records them in the `<INSTALL_DIR>/database/db_tools/logs/Alert_Purge.log file.` (The utility presumes that you have determined the input parameters to specify what matches, alerts, and actions to purge. The utility does not check against the data to verify what it should purge.)

Note: To capture the SQL statements naming set log.diagnostic=true in the `install.cfg.`

The parameters that define what matches to purge consist of one of two possible sets:

- A Behavior Detection job ID, which the `KDD_JOB` table identifies.
- A scenario ID, as defined in the `KDD_SCENARIO` table, and a date range. Behavior Detection does not support multiple scenario IDs so you should run them separately. As part of this input set, you can include an optional comma-separated list of current alert status codes.

The utility then purges actions, then matches, then alerts, according to the contents of the `KDD_AP_ACTION, KDD_AP_MATCH, and KDD_AP_ALERT tables.`

The utility captures purging results and any errors in the `Alert_Purge.log` file.

Note: The Alert Purge Utility does not purge any data from archive tables for erroneous alerts. Also, the system does not update score and previous match count values associated with generated matches and alerts since creation of the erroneous matches.

Automatic Restart Capability

The Alert Purge Utility has an automatic restart capability in that any interruption in the purge processing resumes at that point, regardless of the input parameters. The system documents logs information about the interruption in the `<INSTALL_DIR>/database/db_tools/logs/ Alert_Purge.log` file. Otherwise, any restart that has not progressed to the purge component behaves as a new processing run.

The restart capability allows interrupted purges to resume at a convenient point, but is unable to execute all desired input parameters.

Sample Alert Purge Processes

This section includes three examples of the Purge Alerts process based on input parameters. In these examples, the process executes two jobs: numbers 300000 and 300001, which relate to scenario numbers 300000 and 300001, respectively. As a result of this job, the process creates 50 matches and nine alerts, and performs nine actions.

Table 27 defines the matches that relate to these alerts and actions:

Table 27. Example of Matches and Alerts Associated with Purge Alerts

Match ID Range	Job ID/Scenario ID	Alert ID/Status	Actions/Type/Date
300000-4	300000/300000	None	None
300005-9	300000/300000	300000/OP	None
300010-14	300000/300000	300001/OP	300000 (OP) on 11/6/2006
300015-19	300000/300000	300002/NW	300001 (OP) on 11/6/2005; 300002 on 11/6/2006 (NW)
300020-22	300000/300000	300003/OP	None
300023-24	300001/300001	300003/OP	None
300025-27	300000/300000	300004/OP	300003 (OP) on 11/6/2006
300028-29	300001/300001	300004/OP	300003 (OP) on 11/6/2006
300030-32	300000/300000	300005/NW	300004 (OP) on 11/6/2005 and 300005 on 11/6/2006 (NW)
300033-34	300001/300001	300005/NW	300004 (OP) on 11/6/2005 and 300005 on 11/6/2006 (NW)
300035-39	300001/300001	300006/OP	None
300040-44	300001/300001	300007/OP	300006 (OP) on 11/6/2006
300045-49	300001/300001	300008/NW	300007 (OP) on 11/6/2005; 300008 on 11/6/2006 (NW)

Note: While the Action ID values are not in time-order, their impact on the example above is negligible. The key aspects of the actions relevant to the discussion are the dates of the actions.

As a result, a range of matches is associated either wholly or partly with an alert, and a range of actions taken on the alerts, from either one job and associated scenario, both jobs and their associated scenarios, or the other job and scenario.

The sample Alert Purge Utility output explains the following situations:

- Sample Purge Alerts Utility Run: Situation One shows how to purge those alerts that fully contain the first job in Table 27 (See section *Sample Purge Alerts Utility Run: Situation One*, on page 85 for more information).
- Sample Purge Alerts Utility Run: Situation Two shows how to purge all matches in the first job in Table 27 regardless of their alert affiliation (See section *Sample Purge Alerts Utility Run: Situation Two*, on page 85 for more information).
- Sample Purge Alerts Utility Run: Situation Three explains how to purge only those matches that are generated from scenario 300001 between 11/06/2005 and 11/06/2006, with status OP, and are wholly contained in alerts (See section *Sample Purge Alerts Utility Run: Situation Three*, on page 85 for more information).

Sample Purge Alerts Utility Run: Situation One

To purge only those alerts that contain the first job in Table 27, set the following variables in the `<INSTALL_DIR>/database/db_tools/mantas.cfg/install.cfg` configuration file:

- `job=300000`
- `limit_matches=Y`

This produces the following:

- Matches: 300000-19
- Alerts: 300000-2
- Actions: 300000-2

Sample Purge Alerts Utility Run: Situation Two

To purge all matches in the first job in Table 27, regardless of alert affiliation, set the following variables in the `<INSTALL_DIR>/database/db_tools/mantas.cfg/`

`install.cfg` configuration file:

- `job=300000`
- `limit_matches=N`

This produces the following:

- Matches: 300000-22,300025-27,300030-32
- Alerts: 300000-2
- Actions: 300000-2

Sample Purge Alerts Utility Run: Situation Three

To purge only those matches that scenario 300001 generated between 11/06/2005 and 11/06/2006, with alert status OP, set the following variables in the `<INSTALL_DIR>/database/db_tools/mantas.cfg/install.cfg` configuration file:

- `scenario=300001`
- `start_date='06-Nov-2005'`
- `end_date='06-Nov-2006'`
- `limit_matches=Y`
- `alert_status=OP`

This produces the following results:

- Matches: 300040-44
- Alerts: 300007
- Actions: 300006

Batch Control Utility

The Batch Control Utility enables you to manage and record the beginning and ending of an Behavior Detection batch process. It also enables you to access the currently running batch. You control the process through a job scheduling tool such as Maestro or Unicenter Autosys.

This utility consists of a Java file that resides in the directory `<INSTALL_DIR>/database/db_tools/lib and UNIX script files that reside in <INSTALL_DIR>/database/db_tools/bin:`

- `start_mantas_batch.sh` starts the batch process.
- `end_mantas_batch.sh` ends the batch process.
- `get_mantas_batch.sh` obtains the name of the currently running batch.

The utility also uses common parameters in the configuration file `<INSTALL_DIR>/database/db_tools/mantas_cfg/install.cfg` (See `install.cfg` *file,* on page 66, for more information).

The following sections describe the Batch Control Utility:

- Batches in Behavior Detection
- Directory Structure
- Logs
- Using the Batch Control Utility

Note: To calculate the age in business days versus calendar days, verify that the age.alerts.useBusinessDays setting in the `<INSTALL_DIR>/ database/db_tools/mantas_cfg/install.cfg` file has a value of Y (yes).

Batches in Behavior Detection

Except for the Alert Management subsystem, batches govern all other activity in the Behavior Detection system. A batch provides a method of identifying a set of processing. This includes all activities associated with Data Ingestion and Behavior Detection.

Deployment of a system can be with a single batch.

End-of-day: Represent processing at the completion of a business day for a set of data. Some processes are only appropriate for end-of-day batches. For example, daily activity summary derivations and calculating alert ages are activities that occur only in end-of-day batches. Multiple end-of-day batches per day can run if the Behavior Detection installation supports multiple time zones (for example, New York and Singapore).

Directory Structure

Table 28 provides the directory structure for the Batch Control Utility, in `<INSTALL_DIR>/database/db_tools/`:

Table 28. Batch Control Utility Directory Structure

Directory	Contents
`bin/`	Executable files, including the `start_mantas_batch.sh`, `end_mantas_batch.sh`, and `get_mantas_batch.sh` shell scripts.
`lib/`	Required class files in .jar format.
`mantas_cfg/`	Configuration files (for example, `install.cfg` and `categories.cfg`), in which you can configure properties and logging attributes.
`logs/`	File `batch_control.log` that the utility generates during execution.

Logs

As the Batch control Utility manages batch processing, it generates a date-stamped log in the `<INSTALL_DIR>/database/db_tools/logs/ batch_control.log file`. The log file contains relevant information such as status of various batch control processes, results, and error records.

You can modify the current logging configuration for this utility in the configuration files `<INSTALL_DIR>/database/db_tools/mantas_cfg/ install.cfg` and `categories.cfg`. For more information about logging in these configuration files, See *Common Resources for Administrative Utilities,* on page 66, and Appendix A, *Logging*, on page 133 for more information.

Using the Batch Control Utility

The Batch Control Utility typically runs as part of automated processing that a job scheduling tool such as Maestro or Unicenter AutoSys controls. The utility starts and terminates through a shell script, using values in parameters that particular configuration files contain.

The following sections describe this process, including tasks that you can perform when configuring the utility or running it manually (that is, starting, stopping, or obtaining a batch name).

- Configuring the Batch Control Utility
- Setting Up Batches
- Starting a Batch Process Manually
- Processing for Batch Start
- Ending a Batch Process
- Processing for End Batch
- Identifying a Running Batch Process
- Processing for Obtaining a Batch Name

Configuring the Batch Control Utility

The <INSTALL_DIR>/database/db_tools/mantas_cfg/install.cfg file contains common configuration information that Batch Control and other utilities require for processing (See Figure 27 on page 88). The following sample section from the install.cfg file provides configuration information specific to this utility, including the single parameter that batch control requires.

```
############### BATCH CONTROL CONFIGURATION #####################


# When ending the batch, age alerts in calendar or business days.
age.alerts.useBusinessDays=Y
```

Figure 27. Configuring Batch Control Utility

The value of the age.alerts.useBusinessDays parameter indicates that at completion of an end-of-day batch process, the Behavior Detection application calculates the age of active alerts by number of calendar days (N) or business days (Y). The value of this parameter resides in the `KDD_CAL` table (See Table 36 for more information).

The utility connects to the database employing the user that the `utils.database.username` property specifies in the `install.cfg` file.

Setting Up Batches

Oracle Financial Services delivers with a default batch called DLY. The `KDD_PRCSNG_BATCH` table includes this batch and must contain all batches in the system. When a batch starts as part of an automated process, it uses the batch names and other start-up information in this table.

Table 29 provides the contents of the `KDD_PRCSNG_BATCH` table.

Table 29. KDD_PRCSNG_BATCH Table Contents

Column Name	Description
PRCSNG_BATCH_NM	Name of the batch (for example, DLY).
PRCSNG_BATCH_DSP-LY_NM	Readable name for the batch (for example, Daily).
PRCSNG_ORDER	Relative order of a batch run within processing.
EOD_BATCH_NM	Name of the batch that is this batch's end-of-day. This name is the same as the name for PRCSNG_BATCH_NM if the row represents an end-of-day batch.

Each row in the `KDD_PRCSNG_BATCH` table represents a batch. Each batch identifies the batch that is the corresponding end-of day batch. The following three examples illustrate this concept:

- Single Batch
- Single Site Intra-day Processing
- Multiple Countries

Single Batch

In this example, the `KDD_PRCSNG_BATCH` table contains a single batch per day. This is typical of deployment of a single geography for which a solution set does not require detection more than once daily. The `KDD_PRCSNG_BATCH` table may look similar to the example in Table 30.

Table 30. `Sample KDD_PRCSNG_BATCH` Table with Single Batch

PRCSNG_BATCH_NM	PRCSNG_BATCH_DSPLY_NM	PRCSNG_ORDER	EOD_BATCH_NM
DLY	Daily Batch	1	DLY

Single Site Intra-day Processing

In this intra-day batch example, the system is servicing a single time zone but runs an additional batch during the day to identify behaviors related to overnight trading, as Table 31 describes.

Table 31. `Sample KDD_PRCSNG_BATCH` Table with Intra-day Processing

PRCSNG_BATCH_NM	PRCSNG_BATCH_DSPLY_NM	PRCSNG_ORDER	EOD_BATCH_NM
MAIN	Main Evening Batch	2	MAIN
MORN	Morning Batch	1	MAIN

In this configuration, run the Calendar Manager Utility only during the MORN batch. See *Calendar Manager Utility,* on page 93, for more information. You can run the Data Retention Manager in either the MORN or MAIN batch. If you run it in the MAIN batch, define at least one *buffer* partition so that the MORN batch does not fail due to inadequate partitions.

See *Data Retention Manager,* on page 97, for more information.

Multiple Countries

A single deployment supports detection against data from New York, London, and Hong Kong. In this case, three batches are all end-of-day batches, as Table 32 describes.

Table 32. `Sample KDD_PRCSNG_BATCH` Table with Multiple Country Processing

PRCSNG_BATCH_NM	PRCSNG_BATCH_DSPLY_NM	PRCSNG_ORDER	EOD_BATCH_NM
HK	Hong Kong	1	HK
LND	London	2	LND
NY	New York	3	NY

Since Hong Kong’s markets open first, this is the first batch. You should run the Calendar Manager and Data Retention Manager at the start of the HK batch.

Upon setup of the batches, Behavior Detection processing begins with the `start_mantas_batch.sh` shell script. The final step in a batch is calling the `end_mantas_batch.sh` shell script.

Starting a Batch Process Manually

To start a batch manually, follow these steps:

1. Verify that the Behavior Detection database is operational:

   ```
   tnsping <database instance name>
   ```

2. Verify that the `<INSTALL_DIR>/database/db_tools/mantas_cfg/ install.cfg` configuration file contains the correct source database connection information.
3. Access the directory where the shell script resides:

   ```
   cd <INSTALL_DIR>/database/db_tools/bin
   ```

4. Run the batch control shell script:

   ```
   start_mantas_batch.sh <batch name>
   ```

   where `<batch name>` is the name of the batch. This parameter is case-sensitive.

   If you enter an invalid batch name, the utility terminates and logs a message that describes the error. The error message appears on the console only if you have output to the console enabled in the `<INSTALL_DIR>/database/db_tools/`

   `mantas_cfg/categories.cfg file`. See *Configuring Console Output,* on page 76, for more information.

Processing for Batch Start

After establishing the required Java environment and initiating various Java processing activities, the Batch Control Utility does the following:

1. The utility verifies that the provided batch name contains only the characters A-Z, a-z, and 0-9 by querying the `KDD_PRCSNG_BATCH` table (Table 29).
2. The utility determines whether a batch is running by querying the `KDD_PRCSNG_BATCH_CONTROL` table (Table 33).

Table 33. `KDD_PRCSNG_BATCH_CONTROL` Table Contents

Column Name	Description
PRCSNG_BATCH_ID	Current batch process ID.
PRCSNG_BATCH_NM	Name of the current batch process.
DATA_DUMP_DT	Current business day. The Calendar Manager Utility places this information in the table.
EOD_PRCSNG_BATCH_FL	Flag that indicates whether the batch is an end-of-day process (Y or N).

3. The utility records information about the batch in the `KDD_PRCSNG_BATCH_HIST` table. This table contains a history of all batches that appear by start date and end date.

Table 34 describes the `KDD_PRCSNG_BATCH_HIST` table.

Table 34. `KDD_PRCSNG_BATCH_HIST` Table Contents

Column Name	Description
PRCSNG_BATCH_ID	Current batch process ID.
PRCSNG_BATCH_NM	Name of the current batch process.
DATA_DUMP_DT	Business day on which the batch ran.
START_TS	Time that the batch started.
END_TS	Time that the batch ended (if applicable).
STATUS_CD	Status code that indicates whether the batch is currently running (*RUN*) or has finished (*FIN*).

4. The Batch Control Utility logs a message in the `<INSTALL_DIR>/database/db_tools/logs/batch_control.log` file, stating that the batch process has begun.

Querying the `KDD_PRCSNG_BATCH_HIST` table for confirmation that the batch has started displays information similar to that in Figure 28. In the last entry, note the appearance of `RUN` for `STATUS_CD` and lack of end time in `END_TS`.

```
PRCSNG_BATCH_ID PRCSNG_BATCH_NM   DATA_DUMP_DT    START_TS     END_TS     STATUS_CD
             1   DLY                 10-Nov-06   11-Nov-06  11-Nov-06     FIN
             2   DLY                 11-Nov-06   12-Nov-06  12-Nov-06     FIN
             3   DLY                 12-Nov-06   13-Nov-06  13-Nov-06     FIN
             4   DLY                 13-Nov-06   14-Nov-06  14-Nov-06     FIN
             5   DLY                 14-Nov-06   15-Nov-06  15-Nov-06     FIN
             6   DLY                 15-Nov-06   16-Nov-06  16-Nov-06     FIN
             7   DLY                 16-Nov-06   17-Nov-06  17-Nov-06     FIN
             8   DLY                 17-Nov-06   18-Nov-06  18-Nov-06     FIN
             9   DLY                 18-Nov-06   19-Nov-06  19-Nov-06     FIN
            10   DLY                 19-Nov-06   20-Nov-06  20-Nov-06     FIN
            11   DLY                 20-Nov-06   21-Nov-06                RUN
```

Figure 28. Sample `KDD_PRCSNG_BATCH_HIST` Table—Batch Start Status

Ending a Batch Process

When a batch ends as part of an automated process, the utility retrieves the batch name and other information from the KDD_PRCSNG_BATCH table (See Table 29 on page 88).

To End a Batch Manually

To stop a batch process manually, follow these steps:

1. Verify that the Behavior Detection database is operational.

   `tnsping <database instance name>`

2. Verify that the `<INSTALL_DIR>/database/db_tools/mantas_cfg/ install.cfg` configuration file contains the correct source database connection information.

3. Access the directory where the shell script resides:

   `cd <INSTALL_DIR>/database/db_tools/bin`

4. Start the batch shell script:

   `end_mantas_batch.sh`

If you enter an invalid batch name, the utility terminates and logs a message that describes the error. The error message appears on the console only if you have output to the console enabled in the `<INSTALL_DIR>/database/db_tools/`

mantas_cfg/categories.cfg configuration file.

Processing for End Batch

After establishing the required Java environment and initiating various Java processing activities, the Batch Control Utility does the following:

1. Determines whether a batch is running by querying the `KDD_PRCSNG_BATCH_CONTROL` table (See Table 29 on page 88).
2. Records information about the batch in the `KDD_PRCSNG_BATCH_ HIST` table (See Table 34 on page 91). This table contains a history of all batches that appear by start date and end date. Figure 28 illustrates a sample table query; an end time-stamp in `END_TS` and status of `FIN` in `STATUS_CD` for the bolded entry indicates that the batch has ended.

```
PRCSNG_BATCH_ID  PRCSNG_BATCH_NM   DATA_DUMP_DT    START_TS     END_TS     STATUS_CD
             1   DLY                  10-Nov-06   11-Nov-06  11-Nov-06     FIN
             2   DLY                  11-Nov-06   12-Nov-06  12-Nov-06     FIN
             3   DLY                  12-Nov-06   13-Nov-06  13-Nov-06     FIN
             4   DLY                  13-Nov-06   14-Nov-06  14-Nov-06     FIN
             5   DLY                  14-Nov-06   15-Nov-06  15-Nov-06     FIN
             6   DLY                  15-Nov-06   16-Nov-06  16-Nov-06     FIN
             7   DLY                  16-Nov-06   17-Nov-06  17-Nov-06     FIN
             8   DLY                  17-Nov-06   18-Nov-06  18-Nov-06     FIN
             9   DLY                  18-Nov-06   19-Nov-06  19-Nov-06     FIN
            10   DLY                  19-Nov-06   20-Nov-06  20-Nov-06     FIN
            11   DLY                  20-Nov-06   21-Nov-06  21-Nov-06     FIN
```

Figure 29. KDD_PRSCNG_BATCH_HIST Table-Batch End Status

3. Calculates the age of all open alerts and writes it to `KDD_REVIEW.AGE` if the `EOD_BATCH_FL` is Y in the `KDD_PRCSNG_BATCH_CONTROL` table.
4. Updates the KDD_REVIEW table for all alerts from the current batch to set the Processing Complete flag to Y. This makes the alerts available for alert management.
5. Deletes any records in the KDD_DOC table that the system marks as temporary and are older than 24 hours.
6. Logs a message in the `<INSTALL_DIR>/database/db_tools/logs/ batch_control.log` file, stating that the batch process has begun.

Identifying a Running Batch Process

At times, you may need to know the name of a currently running batch, or verify that a batch is active. For example, during intra-day detection processing, many batches may be running simultaneously and you need to identify one or more by name. To identify a running batch process, use the following procedure.

Caution: If you set the batch control logging to display at the console, be aware that log messages are mixed with the output of the shell script; the output can be difficult to read.

Obtaining a Batch Name

To obtain a batch name, follow these steps:

1. Access the directory where the shell script resides:

   `cd <INSTALL_DIR>/database/db_tools/bin`

2. Start the batch shell script:

   `get_mantas_batch.sh`

The name of the currently running batch is written to standard output (See *Configuring Console Output,* on page 76, for more information).

Processing for Obtaining a Batch Name

After establishing the required Java environment and initiating various Java processing activities, the Batch Control Utility does the following:

1. The utility retrieves the name of the currently running batch from the `KDD_PRCSNG_BATCH_CONTROL` table (See Table 29 on page 88).
2. The utility returns the batch name to standard output.

Calendar Manager Utility

After loading holidays into the `KDD_CAL_HOLIDAY` table and weekly off-days into the `KDD_CAL_WKLY_OFF` table, you can use the Calendar Manager Utility to update and manageBehavior Detection system calendars. You use the utility's Java and shell scripts to connect to the database and perform processing. The `<INSTALL_DIR>/database/db_tools/ mantas_cfg/install.cfg` configuration file contains modifiable inputs that you use to run the utility (See `install.cfg` *file,* on page 66, for more information).

This section contains the following topics:

- Directory Structure
- Logs
- Calendar Information
- Using the Calendar Manager Utility

Directory Structure

Table 35 provides the directory structure for the Calendar Manager Utility, in `<INSTALL_DIR>/database/db_tools/..`

Table 35. Calendar Manager Utility Directory Structure

Directory	Description
`bin/`	Contains executable files, including the shell script `set_mantas_-date.sh`.
`lib/`	Includes required class files in `.jar` format.
`mantas_cfg/`	Contains configuration files (for example, `install.cfg` and `categories.cfg`), in which you can configure properties and logging attributes.
`log/`	Keeps the `calendar_manager.log` log file that the utility generates during execution.

Logs

As the utility updates the calendars in the Behavior Detection system, it generates a log that it enters in the `<INSTALL_DIR>/database/db_tools/logs/calendar_manager.log` file (the logging process time-stamps all entries). The log file contains relevant information such as status of the various Calendar Manager processes, results, and error records.

You can modify the current logging configuration for this utility in the configuration files `<INSTALL_DIR>/database/db_tools/mantas_cfg/install.cfg` and `categories.cfg`. For more information about logging in these configuration files, See *Common Resources for Administrative Utilities,* on page 66, and Appendix A, *Logging*, on page 133, for more information.

Calendar Information

The Calendar Manager Utility obtains all holidays and weekly off-days for loading into the Behavior Detection calendars by retrieving information from the `KDD_CAL_HOLIDAY` and `KDD_CAL_WKLY_OFF` tables (See Table 23 and Table 24). These tables contain calendar information that an Oracle client has provided regarding observed holidays and non-business days.

Using the Calendar Manager Utility

The Calendar Manager Utility runs as part of automated processing that a job scheduling tool such as Maestro or Unicenter AutoSys controls. The utility runs through a shell script, using values in parameters that particular configuration files contain. The utility then populates the `KDD_CAL` database table with relevant Oracle Financial Services business calendar information.

The following sections describe this process, including tasks that you can perform when configuring the utility or running it manually.

- Configuring the Calendar Manager Utility
- Executing the Calendar Manager Utility
- Updating the KDD_CAL Table

Configuring the Calendar Manager Utility

- The `<INSTALL_DIR>/database/db_tools/mantas_cfg/install.cfg` file contains common configuration information that Calendar Manager and other utilities require for processing (See Figure 30). The following sample section from the `install.cfg` file provides configuration information specific to this utility, including default numerical values in the utility's two required parameters.

```
################ CALENDAR MANAGER CONFIGURATION ##################

# The look back and look forward days of the provided date.
# These values are required to update the KDD_CAL table. The
# maximum look back or forward is 999 days.
calendar.lookBack=365
calendar.lookForward=10
```

Figure 30. Calender Manager Configuration

- `calendar.lookBack`: Determines how many days to iterate backward from the provided date during a calendar update.
- `calendar.lookForward`: Determines how many days to iterate forward from the provided date during a calendar update.

The maximum value that you can specify for either of these parameters is 999 days.

Note: The lookback period should be at least 90 days and as long as any alerts are likely to be open. The lookforward period does not need to be more than 10 days. This is used when calculating projected settlement dates during Data Ingestion.

Warning: When you have configured the system to calculate alert age in Business Days, the calendar date of the current system date and the calendar date of the alert creation must be included in the calendar. As such, if you are running with a business date that is substantially behind the current system date, you should set the `lookForward` parameter for the calendar manager sufficiently high to ensure that the system date is included on the calendar. Additionally, if you have alerts that are open for a very long period, you should set the `lookBack` parameter sufficiently high to include the dates of your oldest open alerts. If the business calendar does not cover either of these dates, the processing reverts to calculating age in Calendar days.

The utility connects to the database employing the user that the `utils.database.username` property specifies in the `install.cfg` file.

Executing the Calendar Manager Utility

Typically, you manage the Calendar Manager Utility as part of automated processing. You can run the utility either inside a batch process (that is, after calling the `start_mantas_batch.sh` script) or outside a batch. You can start the utility manually by using the following procedure.

To Start the Utility Manually

To start the Calendar Manager Utility, follow the steps:

1. Verify that the Behavior Detection database is operational:
   `tnsping <database instance name>`
2. Verify that the `<INSTALL_DIR>/database/db_tools/mantas_cfg/install.cfg` configuration file contains the correct source database connection information.
3. Go to the directory where the shell script resides:
   `cd <INSTALL_DIR>/database/db_tools/bin`
4. Start the calendar manager shell script:

```
    set_mantas_date.sh YYYYMMDD
```

where `YYYYMMDD` is the date on which you want to base the calendar (for example, enter November 30, 2006 as *`20061130`*). The utility then verifies that the entered date is valid and appears in the correct format.

If you do not enter a date or enter it incorrectly, the utility terminates and logs a message that describes the error. The error message displays on the console only if you have output to the console enabled in the `<INSTALL_DIR>/database/`
`db_tools/ mantas_cfg/categories.cfg` configuration file. See *Configuring Console Output,* on page 76, for more information.

Updating the KDD_CAL Table

As previously discussed, the Calendar Manager Utility retrieves information that it needs for updating business calendars from the `KDD_CAL_HOLIDAY` and `KDD_CAL_WKLY_OFF` database tables. It then populates the `KDD_CAL` table accordingly. For each calendar name found in the `KDD_CAL_WKLY_OFF` and `KDD_CAL_HOLIDAY` tables, the utility creates entries in `KDD_CAL`.

Table 36 provides the contents of the `KDD_CAL` table.

Table 36. KDD_CAL Table Contents

Column Name	Description
CLNDR_NM	Specific calendar name.
CLNDR_DT	Date in the range between the lookback and lookforward periods.
CLNDR_DAY_AGE	Number of calendar days ahead or behind the provided date. The provided date has age 0, the day before is 1, the day after is –1. For example, if a specified date is 20061129, the CLNDR_DAY_AGE of 20061128 = 1, and 20061130 = –1.
BUS_DAY_FL	Flag that indicates whether the specified date is a valid business day (set the flag to Y). Set this flag to N if the DAY_OF_WK column contains an entry that appears as a valid non-business day in the KDD_CAL_WKLY_OFF table, or a valid holiday in KDD_–CAL_HOLIDAY.
BUS_DAY_AGE	Number of business days ahead or behind the provided date. If BUS_DAY_FL is N, BUS_DAY_AGE receives the value of the previous day's BUS_–DAY_AGE.
BUS_DAY_TYPE_ CD	Indicates the type of business day: ● N = Normal ● C = Closed ● S = Shortened
DAY_OF_WK	Value that represents the day of the week: Sunday=1, Monday=2, Tuesday=3, ... Saturday=7.
SESSN_OPN_TM	Indicates the opening time of the trading session for a shortened day. The format is HHMM.
SESSN_CLS_TM	Indicates the closing time of the trading session for a shortened day. The format is HHMM.
SESSN_TM_OFFST_TX	Indicates the timezone offset for SESSN_OPN_TM and SESSN_CLS_TM. The format is HH:MM.

Table 36. **`KDD_CAL`** Table Contents (Continued)

Column Name	Description
`WK_BNDRY_CD`	Week's start day (SD) and end day (ED). ● If this is the last business day for this calendar name for the week (that is, next business day has a lower DAY_OF_WK value), set to ED<x>, where <x> is a numeric counter with the start/end of the week that the provided date is in = 0. ● If it is the first business day for this calendar name for this week (that is, previous business day has a higher DAY_OF_WK value), set to SD<x>. Weeks before the provided date increment the counter, and weeks after the provided date decrement the counter. Therefore, "ED0" is always on the provided date or in the future, and "SD0" is always on the provided date or in the past.
`MNTH_BNDRY_CD`	Month's start day (SD) and end day (ED). ● If this is the last business day for this calendar name for the month (that is, next business day in a different month), set to ED<y>, where *y* is a numeric counter with the start/end of the month that the provided date is in = 0. ● If it is the first business day for this calendar for this month (that is, previous business day in a different month), set to SD<y>. Months before the provided date increment the counter, and months after the provided date decrement the counter. Therefore, "ED0" is always on the provided date or in the future, and "SD0" is always on the provided date or in the past.

If a batch is running, the system uses the date provided in the call to start the `set_mantas_date.sh` script. This script updates the `KDD_PRSCNG_BATCH_CONTROL.DATA_DUMP_DT` field.

Data Retention Manager

Behavior Detection relies on Oracle partitioning for maintaining data for a desired retention period, providing performance benefits, and purging older data from the database. The data retention period for business and market data is configurable. Range partitioning of the tables is by date.

The Data Retention Manager enables you to manage Oracle database partitions and indexes on a daily, weekly, and/or monthly basis (See Figure 15). This utility allows special processing for trade-related database tables to maintain open order, execution, and trade data prior to dropping old partitions. As administrator, you can customize these tables.

The utility accommodates daily, weekly, and monthly partitioning schemes. It also processes specially configured Mixed Date partitioned tables. The Mixed Date tables include partitions for Current Day, Previous Day, Last Day of Week for weeks between Current Day and Last Day of Previous Month, and Last Business Day of Previous Two Months.

The Data Retention Manager can:

- Perform any necessary database maintenance activities, such as rebuilding global indexes.
- Add and drop partitions, or both, to or from the date-partitioned tables.

Data Retention Manager provides a set of SQL procedures and process tables in the Behavior Detection database. A shell script and a configuration file that contain the various inputs set the environment that the utility uses.

This section covers the following topics:

- Directory Structure
- Logs
- Processing Flow
- Using the Data Retention Manager
- Utility Work Tables

Directory Structure

Table 37 provides the directory structure for the Data Retention Manager.

Table 37. Data Retention Manager Directory Structure

Directory	Contents
`bin/`	Executable files, including the `run_drm_utility.sh` shell script.
`lib/`	Required class files in `.jar` format.
`mantas_cfg/`	Configuration files (for example, `install.cfg` and `categories.cfg`), in which you can configure properties and logging attributes.
`logs/`	File `<INSTALL_DIR>/database/db_tools/logs/ DRM_Utility.log` that the utility generates during execution.

Logs

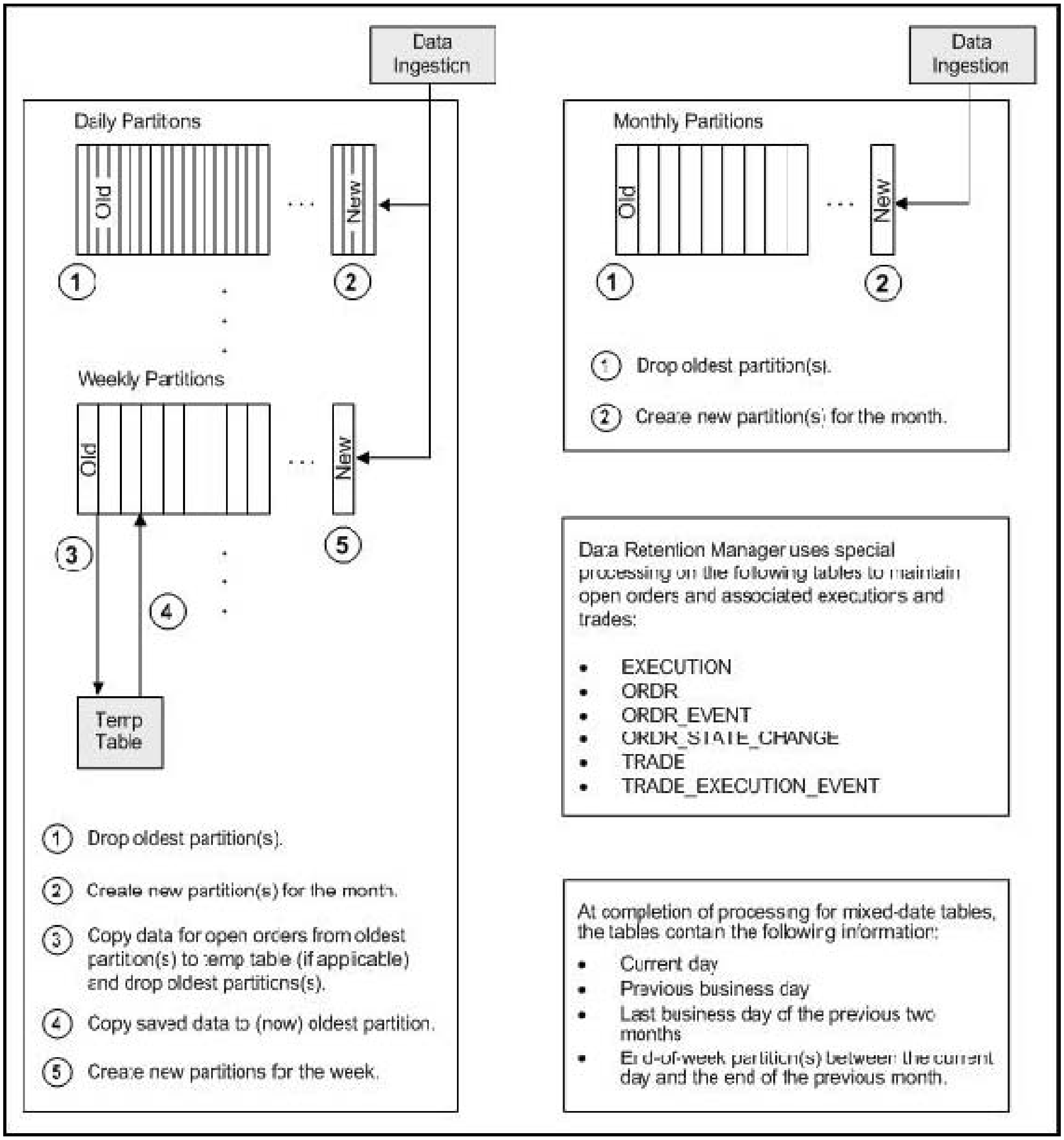
Oracle stored procedures implement Data Retention Manager and conducts some logging on the database server. A configuration parameter in the `install.cfg` file controls the path to which you store the logs on the database server.

As the Data Retention Manager performs partitioning and indexing activities, it generates a log that it enters in the `<INSTALL_DIR>/ database/db_tools/logs/ DRM_Utility.log` file (the logging process time-stamps all entries). The log file contains relevant information such as status of the various processes, results, and error records.

You can modify the current logging configuration for Data Retention Manager in the configuration files `<INSTALL_DIR>/database/db_tools/ mantas_cfg/ install.cfg` and `categories.cfg`. For more information about logging in these configuration files, See *Common Resources for Administrative Utilities,* on page 66, and Appendix A, *Logging*, on page 133 for more information.

Processing Flow

Figure 31 illustrates the Data Retention Manager's process flow for daily, weekly, and monthly partitioning. Based on a table's retention period, the utility drops the oldest partition and then adds a new partition.



① Drop oldest partition(s).

② Create new partition(s) for the month.

③ Copy data for open orders from oldest partition(s) to temp table (if applicable) and drop oldest partitions(s).

④ Copy saved data to (now) oldest partition.

⑤ Create new partitions for the week.

① Drop oldest partition(s).

② Create new partition(s) for the month.

Data Retention Manager uses special processing on the following tables to maintain open orders and associated executions and trades:

- EXECUTION
- ORDR
- ORDR_EVENT
- ORDR_STATE_CHANGE
- TRADE
- TRADE_EXECUTION_EVENT

At completion of processing for mixed-date tables, the tables contain the following information:

- Current day
- Previous business day
- Last business day of the previous two months
- End-of-week partition(s) between the current day and the end of the previous month.

Figure 31. Database Partitioning Process

Using the Data Retention Manager

The Data Retention Manager typically runs as part of automated processing that a job scheduling tool such as Maestro or Unicenter AutoSys controls. However, you can run Data Retention Manager manually on a daily, weekly, or monthly basis to manage database tables. The following sections describe configuration and execution of the utility, and maintain database partitions and indexes.

- Configuring the Data Retention Manager
- Executing the Data Retention Manager
- Creating Partitions
- Maintaining Partitions
- Maintaining Indexes

Configuring the Data Retention Manager

The `<INSTALL_DIR>/database/db_tools/mantas_cfg/install.cfg` file contains common configuration information that Data Retention Manager and other utilities require for processing (See Figure 21 for a sample `install.cfg` file).

Note: The configuration parameters in the `install.cfg` are only used if command line parameters are not provided. It is strongly recommended that you provide command line parameters instead of using the `install.cfg` parameters.

The Data Retention Manager automatically performs system checks for any activity that may result in an error (for example, insufficient space in the tablespace). If it discovers any such activity, it logs a Warning message that identifies the potential problem. If Data Retention Manager fails to run successfully, you can configure the utility so that the ingestion process for the following day still proceeds.

The following sample section from the `install.cfg` file provides other configuration information specific to this utility, including required and optional parameters.

```
######### DATA RETENTION MANAGER CONFIGURATION ##################
# Set the Data Retention Manager input variables here.
##
drm_operation=P
drm_partition_type=A
drm_owner=${schema.mantas.owner}
drm_object_name=A
drm_weekly_proc_fl=Y

#Directory required to put the temporary log from Database Server.
```

Figure 32. Data Retention Manager Configuration

This example shows default values that the system uses only when calling the utility with no command line parameters.

Table 38. Data Retention Manager Processing Parameters

Parameter	Description
`drm_operation`	Operation type: P : Partition AM: Add Monthly Partition DM: Drop Monthly Partition RI: Rebuild Indexes RV:Recompile Views T: Truncate Current Partition
`drm_partition_type`	Partition type: D: Daily W: Weekly M: Monthly X: Mixed-Date A: All Partitions (Daily, Weekly, Monthly)
`drm_owner`	Owner of the object (database schema owner).
`drm_object_name`	Object name. If performing an operation on all objects, the object name is A.
`drm_weekly_proc_fl`	Flag that determines whether partitioning occurs weekly (Y and N).

Note: The system processes Daily partitioned tables (`drm_partition_type=D`) and Mixed-date partitioned tables (`drm_partition_type=X`) simultaneously. Therefore, you need only specify D or X to process these tables.

An example for the Mixed-date partition, for the present date 20050711, is:

```
P20050711 (Current Day)
P20050708 (Previous Day and End of week #1)
P20050701 (End of previous week #2)
P20050630 (End of previous Month #1)
P20050624 (End of previous week #3)
P20050617 (End of previous week #4)
P20050531 (End of previous Month #2)
```

Executing the Data Retention Manager

To execute Data Retention Manager, use the following procedure. Be sure to run the utility when users are not working on the system. To avoid conflicts, Oracle recommends that you use this utility as part of the end-of-day activities.

The Data Retention Manager should be executed nightly for Daily partitioned and Mixed-date partitioned table, after the calendar has been set for the next business day. For weekly and monthly partitioned table, the Data Retention Manager should be executed prior to the end of the current processing period. Oracle recommends running the Data Retention Manager on Thursday or Friday for weekly partitioned tables and on or about the 23rd of each month for monthly partitioned tables.

Note: Be sure to set the system date with the Calendar Manager Utility prior to running the Data Retention Manager (SeeFigure 32, for more information).

Running the Data Retention Manager

To run the Data Retention Manager manually, follow the steps:

1. Access the directory where the shell script resides:
   ```
   cd <INSTALL_DIR>/database/db_tools/bin
   ```
2. Start the batch shell script with the parameters in Data Retention Manager Processing Parameterstable1.fm:
   ```
   run_drm_utility.sh <drm_operation> <drm_partition_type> <drm_owner> <drm_ob-
   ject_name> <drm_weekly_proc_fl>
   ```

Script Examples:

The following are examples of running the script:

- To run the utility for all daily tables in the BUSINESS schema, execute the script:
  ```
  run_drm_utility.sh P D BUSINESS A N
  ```
- To run the utility to drop a monthly partition of the `BUSINESS` table `ACCT_SMRY_MNTH,` execute the script as follows (using the same parameters as in the previous example):
  ```
  run_drm_utility.sh DM M BUSINESS ACCT_SMRY_MNTH N
  ```

Creating Partitions

When creating partition names, use the formats in Table 39

Table 39. Partition Name Formats

Partition Type	Format and Description
Monthly	PYYYYMM where YYYY is the four-digit year and MM is the two-digit month for the data in the partition. For example: Data for November 2006 resides in partition P200611. **Note:** The Data Retention Manager uses information in the `KDD_CAL` table to determine end-of-week and end-of-month boundary dates.
Weekly or Daily	PYYYYMMDD where YYYY is the four-digit year, MM is the two-digit month, and DD is either the date of the data (daily) or the date of the following Friday (weekly) for the data in the partition. For example: Data for November 30, 2006 resides in partition P20061130. Data for the week of November 19 - November 23, 2006 resides in partition P20061123. **Note:** The Data Retention Manager uses information in the `KDD_CAL` table to determine end-of-week and end-of-month boundary dates.

Note: Data Retention Manager assesses the current status of partitions on the specified table to determine the requested partition. If the system previously fulfilled the request, it logs a warning message.

Data Retention Manager does not support multiple partition types on a single table. If an Oracle client wants to alter the partitioning scheme on a table, that client must rebuild the table using the new partitioning scheme prior to utilizing the Data Retention Manager. Then you can update the values in the Data Retention Manager tables to reflect the new partitioning scheme.

Maintaining Partitions

Partition maintenance procedures remove old data from the database so that the database does not continue to grow until space is insufficient. Daily, weekly, or monthly maintenance is necessary for those tables that have daily, weekly, and monthly partitions, respectively.

Partition maintenance:

1. Copies information related to open orders from the oldest partitions to temp tables `(EXECUTION, ORDR, ORDR_EVENT, ORDR_STATE_CHANGE TRADE and TRADE_EXECUTION_EVENT)`
2. Drops the oldest partitions for all partition types.
3. Inserts the saved data into what is now the oldest partition (applicable to tables with open orders).

4. Creates the new partitions.
5. Recompiles the views that scenarios use.

Daily Partitioning Alternative

The Data Retention Manager also enables you to build five daily partitions only a weekly basis rather than daily. You do this by executing the `run_drm_utility.sh` shell script and setting the `drm_weekly_proc_flg` parameter to Y (See Table 38).

This procedure eliminates the need to perform frequent index maintenance; Oracle recommends doing this for large market tables.

This approach builds the daily partitions for the next week. When creating the five daily partitions on a weekly basis, the Data Retention Manager should be executed prior to the end of the current week, to create partitions for the next week.

Note: You must set the `WEEKLY_ADD_FL` parameter in the `KDD_DR_MAINT_OPRTN` table to `Y` so that the procedure works correctly. For more information about this parameter, See Table 38 for more information.

Partition Structures

The structures of business data partitions and market data partitions differ somewhat:

- Business data partitions are predefined so that weekdays (Monday through Friday) are business days, and Saturday and Sunday are *weekly off-days*. Business data tables use all partitioning types.

However, you can use the Calendar Manager Utility to configure a business calendar as desired. For more information about this utility, See *Calendar Manager Utility,* on page 93, for more information.

- Market data partitions hold a single day of data. The partitions use the `PYYYYMMDD` convention, where `YYYYMMDD` is the date of the partition.

Recommended Partition Maintenance

You should run partition maintenance as appropriate for your solution set. Oracle recommends that you run partition maintenance for AML on a daily basis (after setting the business date through the Calendar Manager Utility, and prior to the daily execution of batch processing), and Trading Compliance at least once a week.

Note: Oracle recommends that you use the P (Partition) option when running the Data Retention Manager, as it drops older partitions and adds appropriate partitions in a single run of the utility.

When performing monthly maintenance, you can add or drop a partition independently, as the following procedures describe.

Alternative Monthly Partition Maintenance

As part of an alternative method of monthly partition maintenance, you can either add or drop a monthly database partition, as the following sections describe.

Adding a Monthly Database Partition

To add a monthly partition, run the utility's shell script as follows (See Data Retention Manager Processing Parameterstable1.fm for parameters):

```
run_drm_utility.sh AM M BUSINESS <object> N
```

where `AM` is the `drm_operation` parameter that implies adding a monthly partition.

Dropping a Monthly Database Partition

To drop a monthly partition, run the utility's shell script as follows (See Data Retention Manager Processing Parameterstable1.fm for parameters):

```
run_drm_utility.sh DM M BUSINESS <object> N
```

where, `DM` is the `drm_operation` parameter that implies dropping a partition.

Maintaining Indexes

As part of processing, the Data Retention Manager automatically rebuilds the database index and index partitions that become unusable. You do not need to maintain the indexes separately.

The utility enables you to rebuild global indexes by executing the following command:

```
run_drm_utility.sh RI M BUSINESS <object> N
```

where, RI is the `drm_operation` parameter that implies rebuilding indexes.

Utility Work Tables

The Data Retention Manager uses three work tables during database partitioning, which the following sections describe:

- `KDD_DR_MAINT_OPRTN` Table
- `KDD_DR_JOB` Table
- `KDD_DR_RUN` Table

`KDD_DR_MAINT_OPRTN` Table

The `KDD_DR_MAINT_OPRTN` table contains the processing information that manages Data Retention Manager activities.Table 40 describes the table's contents.

Table 40. BUSINESS.KDD_DR_MAINT_OPRTN Table Contents

Column Name	Description
`PROC_ID`	Identifies the sequence ID for the operation to perform.
`ACTN_TYPE_CD`	Indicates the activity that the utility is to perform on the table: ● A: Analyze ● RI: Rebuild Indexes ● P: Partition ● RV: Recompile Views
`OWNER`	Identifies an owner or user of the utility.
`TABLE_NM`	Identifies a database table.
`PARTN_TYPE_CD`	Indicates the partition type: ● D: Daily ● W: Weekly ● M: Monthly ● X: Mixed Date

Table 40. BUSINESS.KDD_DR_MAINT_OPRTN Table Contents

TOTAL_PARTN_CT	Specifies the total number of partitions to be created, including the current partition. For example, for a daily partitioning scheme of four previous days and the current day, the value of this field is five (5).
BUFFER_PARTN_CT	Specifies the number of buffer partitions the utility is to maintain, excluding the current partition. For example, a two-day buffer has a value of two (2).
CNSTR_ACTN_FL	Determines whether to enable or disable constraints on the table during processing.
WEEKLY_ADD_FL	Indicates whether daily partitions are added for a week at a time. If set to Y, creates Daily Partitions for the next week. For example, if run on a Thursday, the DRM creates the five (5) partitions for the next week beginning with Monday.

Caution: For weekly partitioned tables, do not set the value to Y.

KDD_DR_JOB Table

The KDD_DR_JOB table stores the start and end date and time and the status of each process that the Data Retention Manager calls. Table 41 provides the table's contents.

Table 41. BUSINESS.KDD_DR_JOB Table Contents

Column Name	Description
JOB_ID	Unique sequence ID.
START_DT	Start date of the process.
END_DT	End date of the process.
STATUS_CD	Status of the process: RUN: Running FIN: Finished successfully ERR: An error occurred WRN: Finished with a warning

KDD_DR_RUN Table

The KDD_DR_RUN table stores the start and end date and time and status of individual process runs that are associated with a table. Table 42 describes the table's contents.

Table 42. BUSINESS.KDD_DR_RUN Table Contents

Column Name	Description
JOB_ID	Unique sequence ID.
PROC_ID	Process ID.
START_DT	Start date of the process.
END_DT	End date of the process.

Table 42. BUSINESS.KDD_DR_RUN Table Contents

RSULT_CD	Result of the process: FIN: Finished successfully ERR: An error occurred WRN: Finished with a warning
ERROR_DESC_TX	Description of a resulting error or warning.

The system also uses the `KDD_CAL` table to obtain information such as the dates of the last-day-of-previous-month and end-of-weeks. See Table 36 for contents of the `KDD_CAL` table.

Database Statistics Management

For each of the MANTAS, BUSINESS, and MARKET schemas, the system uses a script to manage Oracle database statistics. These statistics determine the appropriate execution path for each database query.

Logs

The `log.category.RUN_STORED_PROCEDURE` property controls logging for the `process.location` entry in the `<INSTALL_DIR>/database/db_tools/ mantas_cfg/categories.cfg` file.

Using Database Statistics Management

The system calls each script as part of nightly processing at the appropriate time and with the appropriate parameters:

- **MANTAS Schema**: `analyze_mantas.sh [TABLE_NAME] <analysis_type>`
- **BUSINESS Schema**: `analyze_business.sh [TABLE_NAME] <analysis_type>`
- **MARKET Schema**: `analyze_market.sh [TABLE_NAME] <analysis_type>`

The `<analysis_type>` parameter can have one of the following values:

- `DLY_POST_LOAD`: Use this value to update statistics on tables that the system just loaded (for BUSINESS and MARKET schemas).
- **ALL**: Use this once per week on all schemas.
- `DLY_POST_HDC`: Use this value to update statistics of the alert-related archived data (in _ARC tables) in the BUSINESS and MARKET schema tables that the Behavior Detection UI uses to display alerts.
- `DLY_PRE_HDC`: Use this value to update statistics of the Mantas schema tables that contain the alert-related information.

  Note: It is recommended that you do not modify the tables for `DLY_POST_HDC` and `DLY_PRE_HDC`. The Behavior Detection Historical Data Copy procedures use these tables to archive alert-related data.
- `DLY_POST_LINK`: Use this value to update statistics of the Mantas schema tables that contain network analysis information. Run this option at the conclusion of the network analysis batch process.

The `[TABLE_NAME]` parameter optionally enables you to analyze one table at a time. This allows scheduling of the batch at a more granular level, analyzing each table as processing completes instead of waiting for all tables to complete before running the analysis process.

The metadata in the KDD_ANALYZE_PARAM table drive these processes. For each table in the three schemas, this table provides information about the method of updating the statistics that you should use for each analysis type. Valid methods include:

- EST_STATS: Performs a standard statistics estimate on the table.
- EST_PART_STATS: Estimates statistics on only the newest partition in the table.

  Note: For the EST_STATS and EST_PART_STATS parameters, the default sample size that the analyze procedure uses is 5% of the table under analysis. To change the sample percentage, update the SAMPLE_PT column of the desired record in the KDD_ANALYZE_PARAM table.
- IMP_STATS: Imports statistics that were previously calculated. When running an ALL analysis, the system exports statistics for the tables for later use.

  Note: Failure to run the statistics estimates can result in significant database performance degradation.

These scripts connect to the database using the user that the `utils.database.username` property specifies, in the `<INSTALL_DIR>/ database/db_tools/mantas_cfg/install.cfg` file. The `install.cfg` file also contains the following properties:

- `schema.mantas.owner`
- `schema.market.owner`
- `schema.business.owner`

The system derives schema names from these properties.

CHAPTER 7 Administrative Utilities

Oracle Financial Services Behavior Detection Platform provides utilities that enable you to set up or modify a selection of database processes. This chapter describes the Password Manager Utility and the Scenario Threshold Editor.

Password Manager Utility

To change a password in any subsystem other than alert management and admin tools, execute the command:

`<INSTALL_DIR>/changePassword.sh`: This prompts for the passwords of all the required application users. The passwords that are entered are not output to (that is, not shown on) the screen and the same password must be re-entered in order to be accepted. If it is not necessary to change a given password, press the Enter key to skip to the next password. The password that was skipped was not changed. The script prompts for passwords for the following users, depending on what subsystems have been installed:

- Data Ingest User
- Database Utility User
- Algorithm User
- Data Miner User

If there is a need to change a specific password property stored in an application configuration file, the following command can be run:

```
<INSTALL_DIR>/changePasswords.sh <property name>
```

For example,

```
<INSTALL_DIR>/changePasswords.sh email.smtp.password
```

Note: If you are running this utility for the first time after installation, execute the command as specified below. Note that all passwords must be entered and it is not possible to skip a password.

```
<INSTALL_DIR>/changePassword.sh all
```

For changing password for admin tools subsystem, execute the command `FIC_HOME/AM/changePassword.sh`.This prompts for the passwords of the following users:

- Web Application User
- Data Miner User

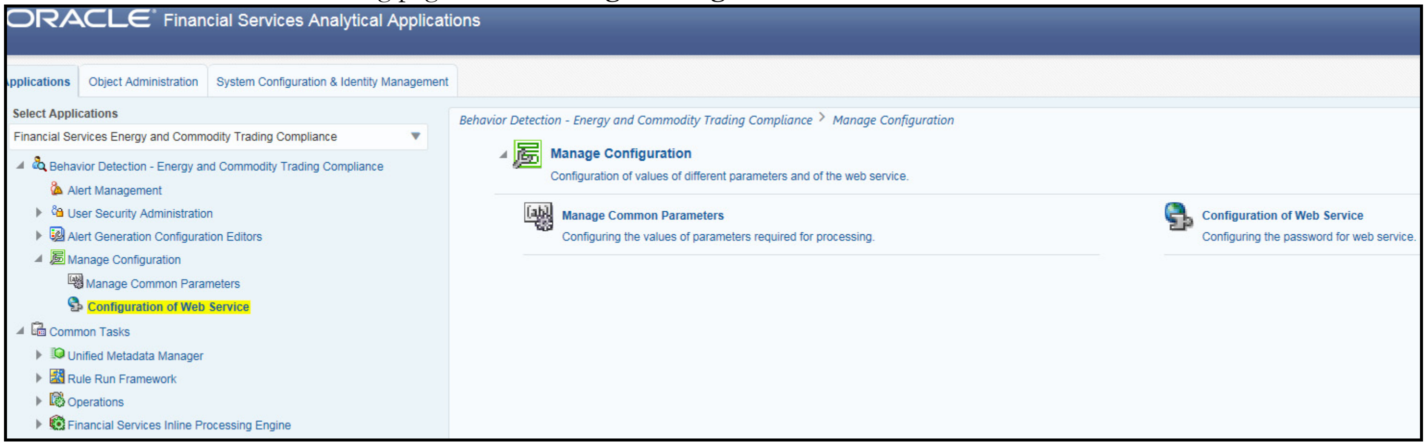
When changing a password for the admin tools subsystem, if the Web application is deployed from a WAR file, the WAR file must be regenerated by running `FIC_HOME/AM/create_at_war.sh`.

Configuring the Exemption Password

The Exemption Web Service password can be changed using the Encrypt utility through the Web Service configuration UI. To change this password, follow these steps:

Log in as an Admin user who is mapped to the MANTAS user group.

1. In the OFSAAI landing page, select **Manage Configuration.**


ORACLE Financial Services Analytical Applications

pplications | Object Administration | System Configuration & Identity Management

Select Applications
Financial Services Energy and Commodity Trading Compliance

- Behavior Detection - Energy and Commodity Trading Compliance
  - Alert Management
  - User Security Administration
  - Alert Generation Configuration Editors
  - Manage Configuration
    - Manage Common Parameters
    - Configuration of Web Service
- Common Tasks
  - Unified Metadata Manager
  - Rule Run Framework
  - Operations
  - Financial Services Inline Processing Engine

Behavior Detection - Energy and Commodity Trading Compliance > Manage Configuration

Manage Configuration
Configuration of values of different parameters and of the web service.

Manage Common Parameters
Configuring the values of parameters required for processing.

Configuration of Web Service
Configuring the password for web service.

Figure 33. Manage Configuration options.

2. Select **Configuration of Web Services**. The Encrypt Utility opens.
3. Enter the new password in the **Enter Password for CTR Exemption Web Service** field. Click **Encrypt**. The encrypted password is updated in the kdd_install_param for the param_id=102 (WEB SERVICE PARAMETERS). This password can be used for the SOAP request with the user provided in the attr_1_value_tx column in the kdd_install_param table.

Scenario Threshold Editor

When scenarios are created, thresholds are established that enable you to modify the values of these thresholds in a production environment. Once the application is in the production environment, any user assigned the CTR Admin user group can use the Scenario Threshold Editor to modify threshold values of any installed scenario, and threshold sets to fine-tune how that scenario finds matches. Using this tool, you can enter a new value for a threshold (within a defined range) or reset the thresholds to their sample values.

The Scenario Editor threshold page can be used for modifying the scenario thresholds and test run the scenario to know the number of matches that are generated through the test run. It can also be used to create a new threshold set based on the already available threshold set to modify the threshold and test the scenario.

A scenario is installed using the sample list of thresholds and values. This sample list of thresholds is referred to as the *base threshold set*. During deployment, you can create additional threshold sets to support specific business needs using the Oracle Financial Services Scenario Manager application.

Note: Changing scenario threshold values can generate significantly more or less alerts, depending upon the modifications made.

The following subsections discuss features you encounter while using the Scenario Threshold Editor:

- Threshold Sets
- Inactive Thresholds

For more information about scenarios, refer to the respective Technical Scenario Description document (for example, for trading compliance scenario information, refer to the *Trading Compliance Technical Scenario Descriptions).*

Threshold Sets

Threshold sets allow you to run the same scenario multiple times against a variety of sources (for example, exchanges, currencies, or jurisdictions) with separate threshold values for each source.

For example, you may have a scenario with the base threshold set and two additional threshold sets that were created during deployment. You decide that you need this scenario to detect matches in transactions with a minimum value in US currency, European currency, and Japanese currency. Rather than changing the base threshold set for each situation, you can set the value of the base threshold set to detect US currency (for example, USD 100,000), the second threshold set to detect European currency (for example, EUR 150,000), and the third threshold set to detect Japanese currency (for example, JPY 125,000).

Since threshold sets two and three have only a few fields that differ from the base threshold set, you can check the Inherit Base Value check box feature for those fields that are exactly the same as the base threshold set. This feature associates the threshold values in the threshold set you are modifying with the corresponding values in the base threshold set. This association copies the corresponding base threshold set values to the set you are modifying and automatically updates them if the base value changes (refer to *<Scenario–Threshold Set> Area*, on page 115 for more information).

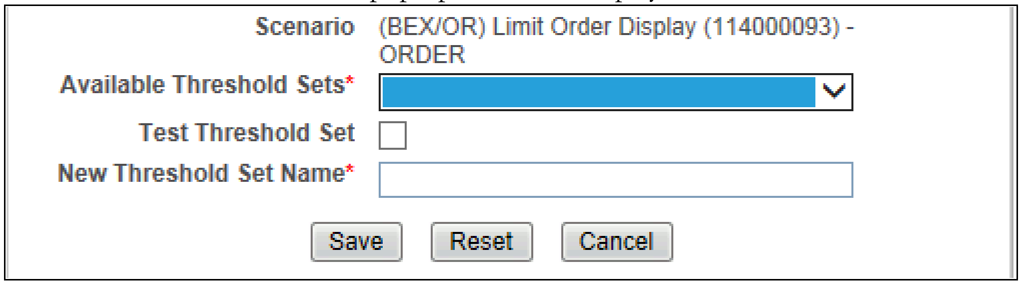
You do not have to run all three jobs all the time. Each threshold set has a unique ID, so you can tell the system which set to run and how often to run it. Refer to your scheduling tool's (for example, Control-M) documentation to sequence these jobs.

Note: Use the Scenario Threshold Editor to modify the values of existing threshold sets. Threshold sets can be created either through the add new threshold set button or through the Scenario Manager.

Adding a New Threshold Set

To add a new Threshold Set, follow these steps:

1. Under the **Administration** menu, hover over **Behavior Detection - Currency Transaction Reporting**, click **Alert Generation Configuration Threshold Editor.**
2. Select the scenario from the scenario drop-down list.
3. Click **Add New Threshold Set**.
4. The *Add New Threshold Set* pop-up window is displayed.



Scenario	(BEX/OR) Limit Order Display (114000093) - ORDER
Available Threshold Sets*	
Test Threshold Set	☐
New Threshold Set Name*	

Figure 34. Add new Threshold Set window

5. Enter the required details in the following fields:

Table 43. Add New Threshold Set Components

Field	Description
Scenario	This field is non editable and displays the scenario that has been selected in the drop-down list from the threshold editor page.
Available Threshold Sets	This dropdown box displays all the available threshold sets in the system for the selected scenario. Also, this is required to acquire the thresholds for the new threshold set that is being created. **Note :** ● If the user do not select a value from the “Available Threshold sets” drop-down list, the following error message is displayed: *Please select a threshold set from the available threshold sets dropdown to create a new threshold set.* ● If the user has selected a threshold set which doesn't have an associated job, then the following error message is message: *The selected threshold set doesn’t have the required Job and Job Dataset for running the scenario test execution. Please select any other threshold set and take action.*
Test Threshold Set	Select this checkbox if the threshold set created is a test threshold set or not. The threshold set name is available `threshold set name + _TST_datetimestamp`.
New Threshold Set Name	By default, this field is kept blank. You can enter the threshold set name only when the *Create Test Threshold Set* checkbox is not selected. When the user has selected the Test Threshold Set checkbox, then this textbox is pre populated with a value that has been selected from the available threshold sets along with a time stamp.

6. Click **Save**. The Threshold Set is added.

Delete Test Threshold Set

To delete a test Threshold Set, follow these steps:

1. Under the **Administration** menu, hover over **Alert Management Configuration**, click **Threshold Editor.**
2. Click **Delete Test Threshold Set**.
3. The Threshold Set is deleted.

Inactive Thresholds

For scenarios to work properly, thresholds that are not being used by a scenario must have their values set to Inactive. The following groups of thresholds can have values set to Inactive:

- Mutually Exclusive Thresholds
- Additional Scenario Thresholds

Mutually Exclusive Thresholds

In some situations, scenarios apply the value of one threshold only when the value of another threshold is set to *N* for no. These types of thresholds are referred to as a *mutually exclusive* thresholds.

For example, the use of the *Included Jurisdiction Codes* threshold is contingent upon the value of the *All Jurisdictions* threshold.

Table 44 shows how mutually exclusive thresholds work in two different situations.

Table 44. Mutually Exclusive Thresholds

Threshold	Situation 1	Situation 2
All Jurisdictions	Y	N
Included Jurisdiction Codes	Inactive	North, East

If the value of the *All Jurisdictions* threshold is set to Y for yes (Situation 1), then the *Included Jurisdiction Codes* threshold values are not used and have the value set to Inactive. Conversely, if the value of the *All Jurisdictions* threshold is set to *N* for no (Situation 2), then the scenario only uses the value specified by the *Included Jurisdiction Codes* threshold (that is, North, East).

Additional Scenario Thresholds

Your deployment may not need to utilize all the thresholds established within a particular scenario. The mutually exclusive thresholds not used by the scenario are set to Inactive.

About the Scenario Threshold Editor Screen Elements

The following screen elements display in the Scenario Threshold Editor:

- Search Bar
- <Scenario–Threshold Set> Area

Search Bar

The search bar allows you to search for threshold values by selecting a specific scenario and threshold set (Figure 35).

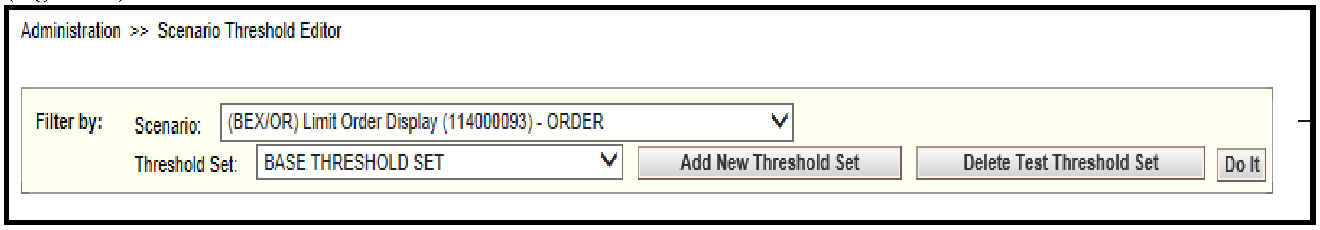

Administration >> Scenario Threshold Editor
Filter by:
Scenario:
(BEX/OR) Limit Order Display (114000093) - ORDER
Threshold Set:
BASE THRESHOLD SET
Add New Threshold Set
Delete Test Threshold Set
Do It

Figure 35. Search Bar

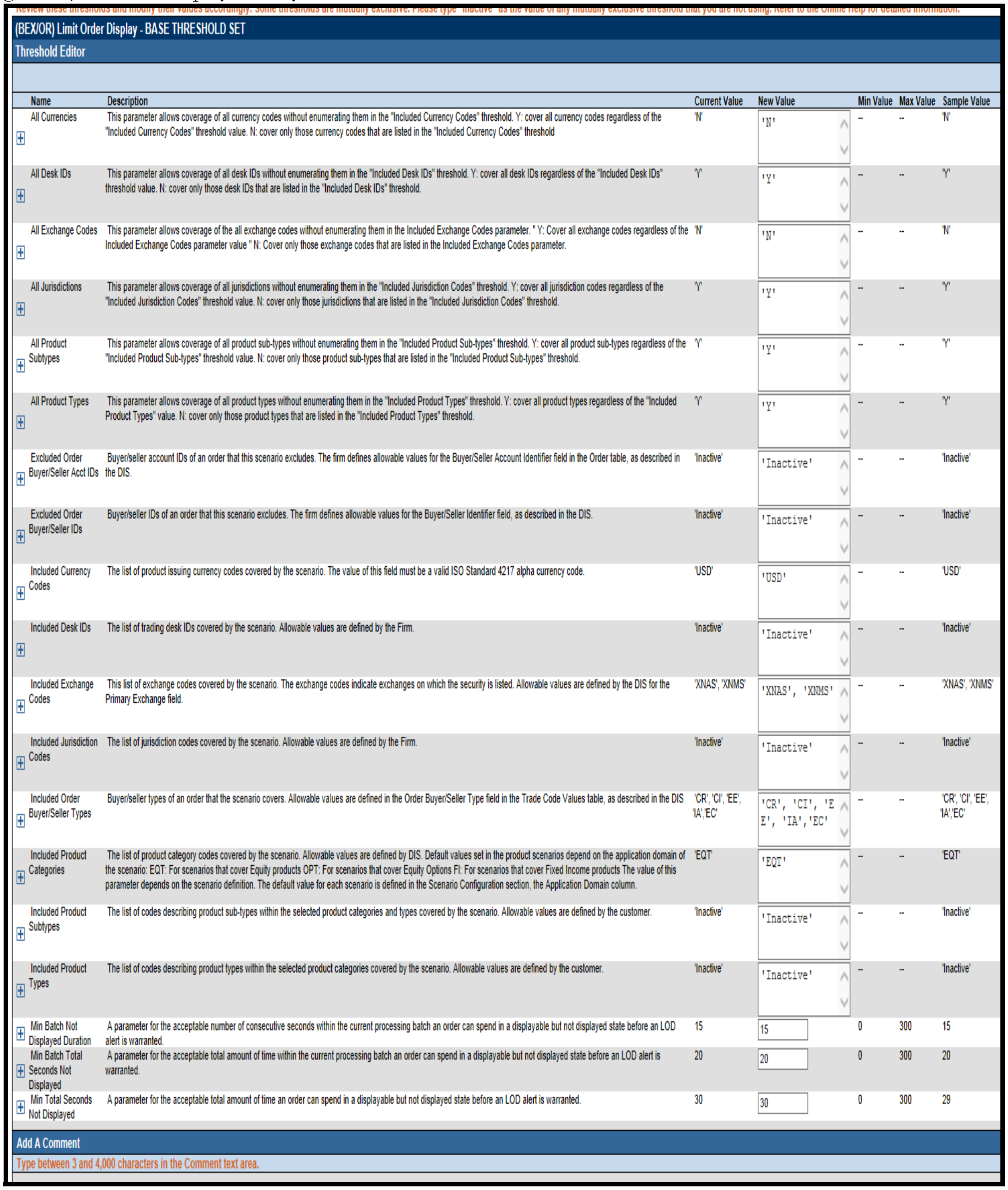
The components of the search bar includes the following:

- **Filter by: Scenario** drop-down list: Provides a list of scenarios displayed by the scenario's short name, ID number, and focus type (for example, Avoid Report Thresh (106000129) – ACCOUNT).
- **Filter by: Threshold Set** drop-down list: Provides a list of Threshold Sets associated with the scenario displayed in the Scenario drop-down list. The base threshold set displays first, followed by additional threshold sets listed in ascending alphabetical order.
- **Add New Threshold Set**: When clicked, enables you to add new threshold sets.
- **Delete Test Threshold Set:** When clicked, enables you to delete test threshold sets.

- **Do It** button: When clicked, displays the threshold values for the scenario and threshold set selected in the search bar.

<Scenario–Threshold Set> Area

The <Scenario-Threshold Set> Area displays the list of threshold values for a selected scenario and threshold set (Figure 36). This list displays after you select a scenario and threshold set in the search bar and click **Do It**.



Review these thresholds and modify their values accordingly. Some thresholds are mutually exclusive. Please type 'Inactive' as the value of any mutually exclusive threshold that you are not using. Refer to the Online Help for detailed information.

(BEX/OR) Limit Order Display - BASE THRESHOLD SET

Threshold Editor

Name	Description	Current Value	New Value	Min Value	Max Value	Sample Value
All Currencies	This parameter allows coverage of all currency codes without enumerating them in the "Included Currency Codes" threshold. Y: cover all currency codes regardless of the "Included Currency Codes" threshold value. N: cover only those currency codes that are listed in the "Included Currency Codes" threshold	'N'	'N'	--	--	'N'
All Desk IDs	This parameter allows coverage of all desk IDs without enumerating them in the "Included Desk IDs" threshold. Y: cover all desk IDs regardless of the "Included Desk IDs" threshold value. N: cover only those desk IDs that are listed in the "Included Desk IDs" threshold.	'Y'	'Y'	--	--	'Y'
All Exchange Codes	This parameter allows coverage of the all exchange codes without enumerating them in the Included Exchange Codes parameter. " Y: Cover all exchange codes regardless of the Included Exchange Codes parameter value " N: Cover only those exchange codes that are listed in the Included Exchange Codes parameter.	'N'	'N'	--	--	'N'
All Jurisdictions	This parameter allows coverage of all jurisdictions without enumerating them in the "Included Jurisdiction Codes" threshold. Y: cover all jurisdiction codes regardless of the "Included Jurisdiction Codes" threshold value. N: cover only those jurisdictions that are listed in the "Included Jurisdiction Codes" threshold.	'Y'	'Y'	--	--	'Y'
All Product Subtypes	This parameter allows coverage of all product sub-types without enumerating them in the "Included Product Sub-types" threshold. Y: cover all product sub-types regardless of the "Included Product Sub-types" threshold value. N: cover only those product sub-types that are listed in the "Included Product Sub-types" threshold.	'Y'	'Y'	--	--	'Y'
All Product Types	This parameter allows coverage of all product types without enumerating them in the "Included Product Types" threshold. Y: cover all product types regardless of the "Included Product Types" value. N: cover only those product types that are listed in the "Included Product Types" threshold.	'Y'	'Y'	--	--	'Y'
Excluded Order Buyer/Seller Acct IDs	Buyer/seller account IDs of an order that this scenario excludes. The firm defines allowable values for the Buyer/Seller Account Identifier field in the Order table, as described in the DIS.	'Inactive'	'Inactive'	--	--	'Inactive'
Excluded Order Buyer/Seller IDs	Buyer/seller IDs of an order that this scenario excludes. The firm defines allowable values for the Buyer/Seller Identifier field, as described in the DIS.	'Inactive'	'Inactive'	--	--	'Inactive'
Included Currency Codes	The list of product issuing currency codes covered by the scenario. The value of this field must be a valid ISO Standard 4217 alpha currency code.	'USD'	'USD'	--	--	'USD'
Included Desk IDs	The list of trading desk IDs covered by the scenario. Allowable values are defined by the Firm.	'Inactive'	'Inactive'	--	--	'Inactive'
Included Exchange Codes	This list of exchange codes covered by the scenario. The exchange codes indicate exchanges on which the security is listed. Allowable values are defined by the DIS for the Primary Exchange field.	'XNAS', 'XNMS'	'XNAS', 'XNMS'	--	--	'XNAS', 'XNMS'
Included Jurisdiction Codes	The list of jurisdiction codes covered by the scenario. Allowable values are defined by the Firm.	'Inactive'	'Inactive'	--	--	'Inactive'
Included Order Buyer/Seller Types	Buyer/seller types of an order that the scenario covers. Allowable values are defined in the Order Buyer/Seller Type field in the Trade Code Values table, as described in the DIS	'CR', 'CI', 'EE', 'IA','EC'	'CR', 'CI', 'EE', 'IA','EC'	--	--	'CR', 'CI', 'EE', 'IA','EC'
Included Product Categories	The list of product category codes covered by the scenario. Allowable values are defined by DIS. Default values set in the product scenarios depend on the application domain of the scenario: EQT: For scenarios that cover Equity products OPT: For scenarios that cover Equity Options FI: For scenarios that cover Fixed Income products The value of this parameter depends on the scenario definition. The default value for each scenario is defined in the Scenario Configuration section, the Application Domain column.	'EQT'	'EQT'	--	--	'EQT'
Included Product Subtypes	The list of codes describing product sub-types within the selected product categories and types covered by the scenario. Allowable values are defined by the customer.	'Inactive'	'Inactive'	--	--	'Inactive'
Included Product Types	The list of codes describing product types within the selected product categories covered by the scenario. Allowable values are defined by the customer.	'Inactive'	'Inactive'	--	--	'Inactive'
Min Batch Not Displayed Duration	A parameter for the acceptable number of consecutive seconds within the current processing batch an order can spend in a displayable but not displayed state before an LOD alert is warranted.	15	15	0	300	15
Min Batch Total Seconds Not Displayed	A parameter for the acceptable total amount of time within the current processing batch an order can spend in a displayable but not displayed state before an LOD alert is warranted.	20	20	0	300	20
Min Total Seconds Not Displayed	A parameter for the acceptable total amount of time an order can spend in a displayable but not displayed state before an LOD alert is warranted.	30	30	0	300	29

Add A Comment

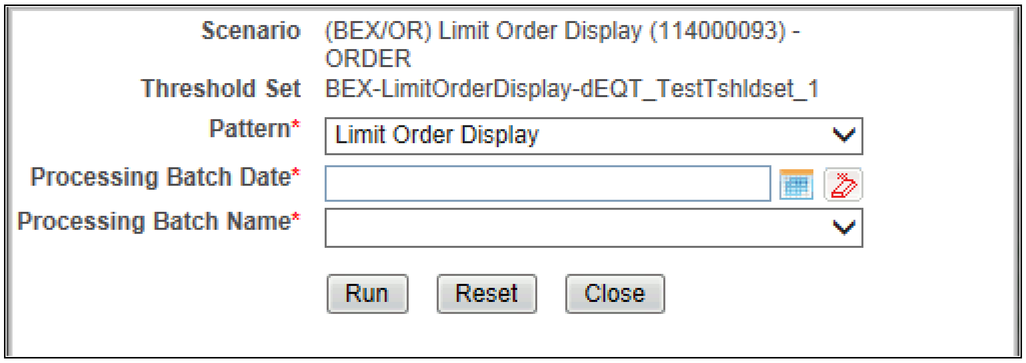
Type between 3 and 4,000 characters in the Comment text area.

Figure 36. <Scenario-Threshold Set> Area

The <Scenario-Threshold Set> Area includes the following components and contents:

- Long name of the scenario and the name of the threshold set in the title of the <Scenario-Threshold Set> bar.
- List of scenario thresholds by threshold name, sorted in ascending alphabetical order.
- Threshold information as follows:
  - **Threshold History Icon**: Expands or contracts the Threshold History inset that displays a history of all modifications to the selected threshold value in reverse chronological order by creation date. Information displayed includes the creation date, user name, threshold value, and any comment associated with the threshold value change.

  If comments are displayed and the comment text consists of more than 100 characters, the Scenario Threshold Editor displays the first 100 characters followed by an ellipsis (...) indicating that more text is available. When you click the ellipsis, the entire comment displays in the Expanded Comments dialog box for ease of viewing.
  - **Name**: Displays the name of the threshold.
  - **Description**: Displays the description of the threshold.
  - **Current Value**: Displays the current value of the threshold. If the data type of the threshold is *LIST*, multiple values are displayed in a comma-delimited list, with each value contained in single quotes (' '). Thresholds with an *Inactive* current value are not being used by the scenario (refer to *Inactive Thresholds*, on page 112 for more information).
  - **Inherit Base Value**: Enables you to select the check box to apply the corresponding threshold values from the base threshold set to the threshold set displayed. Selecting the check box disables the New Value text box. This option does not display for the base threshold set.
  - **New Value**: Displays the current value of the threshold in the editable New Value text box if the Inherit Base Value check box is not selected. If the data type for the threshold is *LIST*, multiple values are displayed in a comma-delimited list, with each value contained in single quotes (' ').
  - **Min Value**: The minimum value of the threshold.
  - **Max Value**: The maximum value of the threshold.
  - **Sample Value**: The sample value of the threshold.
  - **Data Type**: The type of data that is utilized by a threshold in a scenario. There are five data types: Integer, Boolean, Real, String, and List. Place your cursor over this value to display the threshold unit of measure (for example, days, percentage, or distance).
  - **Add A Comment**: Provides a place to type comments. When you type a comment and click **Save**, the same comment is applied to each modified threshold.
- **Restore Samples Values:** Restores all thresholds within the selected scenario threshold set to the sample values
- **Save**: Saves all modifications to the database.
- **Cancel**: Redisplays the Scenario Threshold Editor without the <Scenario-Threshold Set> Area and does not save your changes.
- **Test:** When the Test button is clicked**,** *Scenario Test Execution* pop-up window is displayed.



Scenario	(BEX/OR) Limit Order Display (114000093) - ORDER
Threshold Set	BEX-LimitOrderDisplay-dEQT_TestTshldset_1
Pattern*	Limit Order Display
Processing Batch Date*	
Processing Batch Name*	

Figure 37. Scenario Test Execution window

The Scenario Test Execution window displays the following fields:

Table 45. Scenario Test Execution components

Field	Description
Scenario Name	This field is non-editable and displays the scenario that has been selected in the drop-down list from the threshold editor page.
Threshold Set	This is a non-editable text box which displays the threshold set name that has been selected for test run.
Pattern	Select the pattern from the drop-down list that are part of the selected scenario. **Note:** Since the scenario job runs based on the pattern, you cannot run multiple patterns of the scenario at the same time.
Processing Batch Date	Select the date based on which the scenario patterns will run.
Processing Batch Name	Select the batch name from the drop-down list. **Note:** If a Batch with the selected Processing Batch Name and Date is already running, then the following error message is displayed: *A Batch with the selected Processing Batch Name and Date is already running. Please wait till the Batch completes*.

After selecting these values, click the **Run** button to run the scenario and the results can be viewed in the *Review Test Scenario Results* page.

- **Update Product Threshold Set**: Enables you to update the test threshold set to product threshold set. This button is enabled only when the threshold set selected is newly created threshold.

Using the Scenario Threshold Editor

The Scenario Threshold Editor configures scenario threshold values by:

- Providing threshold values for a specific scenario and threshold set
- Accepting and validating user-entered threshold values
- Saving the modified threshold values to the database

This section explains the following functions of the Scenario Threshold Editor:

- Changing a Scenario Threshold
- Resetting a Scenario Threshold to the Sample Values

- Viewing a Scenario Threshold's History
- Viewing Expanded Comments

Changing a Scenario Threshold

To change a scenario threshold value, follow these steps:

1. Select the desired scenario from the **Filter by: Scenario** drop-down list.
2. Select the desired threshold set from the **Filter by: Threshold Set** drop-down list.
3. Click **Do It**.

   The system displays the threshold values for the scenario and threshold set selected.
4. Type a new value in the **New Value** box for each threshold that you wish to update.

   If you are not updating a base threshold set, you can inherit corresponding values from the base threshold set by checking the **Inherit Base Value** check box.

   Optional: Enter any comments in the **Add A Comment** text box.
5. Click **Save**.

   The new threshold values display in the Threshold List for <Scenario-Threshold Set>.

Resetting a Scenario Threshold to the Sample Values

To reset a scenario's threshold sample values, follow these steps:

1. Select the desired scenario from the **Filter by: Scenario** drop-down list.
2. Select the desired threshold set from the **Filter by: Threshold** Set drop-down list.
3. Click **Do It**.

   The system displays the threshold values for the scenario and threshold set selected.
4. Click **Restore Sample Values** button.

   The Confirmation dialog box displays the following message: *Are you sure you want to restore the threshold values of the displayed threshold set to their sample values?*

   To restore thresholds that have the Inherit Base Value check box selected, you must clear the check box. Click **OK** to return to the Threshold Editor with the sample values displayed, then click **Save.** Click **Cancel** to retain the current values.
5. Click **OK**.

   The dialog box closes and the sample values display in the [Scenario-Threshold Set] Area.
6. Click **Save**.

   The database is updated to reflect the changes.

Viewing a Scenario Threshold's History

To view the modification history for a specific threshold, follow these steps:

1. Click **Expand** next to the desired threshold.

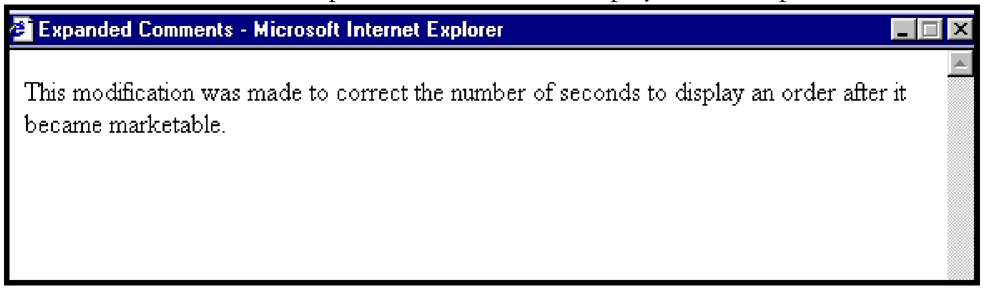
   The Threshold History inset displays with the history for the threshold selected.
2. Click **Contract** next to the threshold to hide the Threshold History inset.

Viewing Expanded Comments

To view an expanded comment in the Scenario Threshold inset, follow these steps:

1. Click the **ellipsis (...)** at the end of the comment in the Scenario Threshold inset.

   The entire comment, up to 4,000 characters, displays in the Expanded Comments dialog box (Figure 38).



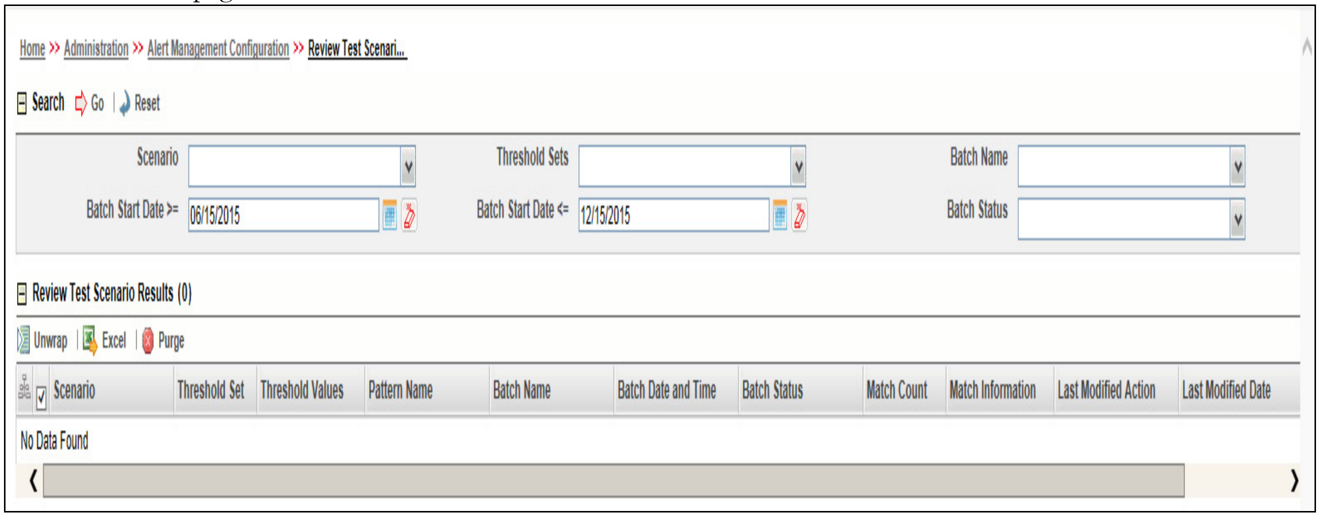
This modification was made to correct the number of seconds to display an order after it became marketable.

Figure 38. Example Expanded Comment Dialog Box

2. Click **X (Close button)** on the top right corner to close the dialog box.

Review Test Scenario Results

The Review Test Scenario Results page allows you to review the results of the test scenarios that are run from the threshold editor page.


Home >> Administration >> Alert Management Configuration >> Review Test Scenari...
Search
Go
Reset
Review Test Scenario Results (0)
Unwrap
Excel
Purge

Scenario		Threshold Sets	
Batch Start Date >=	06/15/2015	Batch Start Date <=	12/15/2015
Batch Name		Batch Status	

Scenario	Threshold Set	Threshold Values	Pattern Name	Batch Name	Batch Date and Time	Batch Status	Match Count	Match Information	Last Modified Action	Last Modified Date
No Data Found										

Figure 39. Review Scenario Details page

About the Review Test Scenario Details Screen Elements

The following screen elements display in the review scenario details:

- "Search Bar"
- "Search and List Grid"

Search Bar

The components of the search bar includes the following:

Table 46. Search Bar Components

Field	Description
Scenario Name	Select the scenario name from the drop-down list.
Threshold Sets	Select the threshold set from the drop-down list. **Note:** This field will display the value only when the scenario name is selected. When no scenario name is selected, then it is assumed that all the threshold sets are included for the search.
Batch Name	Select the Batch name on which the scenario is run. **Note:** By default, this field is kept as blank. This fetches value only when the scenario name is selected.
Batch Start Date >=	Select the current date from the calendar. By default, this column will display Current Date - 6months.
Batch Start Date <=	Select the current date from the calendar. By default, this column will display Current Date
Batch Status	Select the batch status from the drop-down list. Following are the options available: ● Running ● Finished ● Error ● Cancelled

Search and List Grid

The search results are displayed in the *Search and List* grid, which displays the following columns:

Table 47. Search and List grid components

Field	Description
Scenario	This field displays the scenario name, as displayed in the Threshold Editor Page.
Threshold Set	This field displays the threshold set name, as displayed in the Threshold Editor Page.
Threshold Value	This field displays the threshold value, as displayed in the Alert Details Page.
Pattern Name	This field displays the pattern name for the selected pattern ID.
Batch Name	This field displays the batch name on which the scenario is run.
Batch Date and Time	This field displays the batch date and time.
Batch Status	This field displays the batch status. Following is the batch status: ● Running ● Finished ● Error ● Cancelled

Table 47. Search and List grid components

Field	Description
Match Count	This field displays the number of match generated for the run.
Match Information	This field displays an excel icon. When clicked, an auto generated spreadsheet in the format "`<Scenario Name>_<BatchDate>_Matched Details.xlsx`" is displayed. The details that are mentioned in <> are dynamic based on the scenario selection.
Last Modified Action	This field displays the action that was last modified.
Last Modified Date	This field displays the date on which the last action has been taken..
Last Modified User	This field displays the user who have taken the last action.

Action Buttons

The following action button is available on the *Search and List* grid:.

- **Purge:** When clicked**,** this button purges all the matches that are generated for the selected scenario and threshold set combination.

CHAPTER 8 CTR Batch Execution

CTR Reports can be generated through a batch process that can be executed periodically such as Daily, Weekly, Monthly, Quarterly, and Half-yearly depending on an organization's requirement.

You can configure the CTR Reports batches as per the business process requirements of the organization. The OFS CTR has the CTR batch that assesses accounts and create Individual and Pooled CTR Reports.

This chapter details the configuration of Batches and includes the steps for the following:

- Scheduling a Batch
- Monitoring a Batch After Execution
- Cancelling a Batch after Execution
- Re-starting a Batch
- Re-running a Batch
- Batch Tasks

Scheduling a Batch

Ensure all the required servers, that is, ICC, Router, and Message should be up and running before executing a batch. For more information on starting servers, See the *Oracle Financial Services Analytical Applications Infrastructure Installation and Configuration Guide.*

When an organization wants to run the batches periodically, a CTR Administrator user can schedule the batches to run either once, daily, weekly, or months.

Note: Before scheduling a batch, ensure that the ICC router and message server are up and running. For more information on starting ICC router and message server, see the *Oracle Financial Services Analytical Applications Infrastructure User Manual Release 8.0.2*

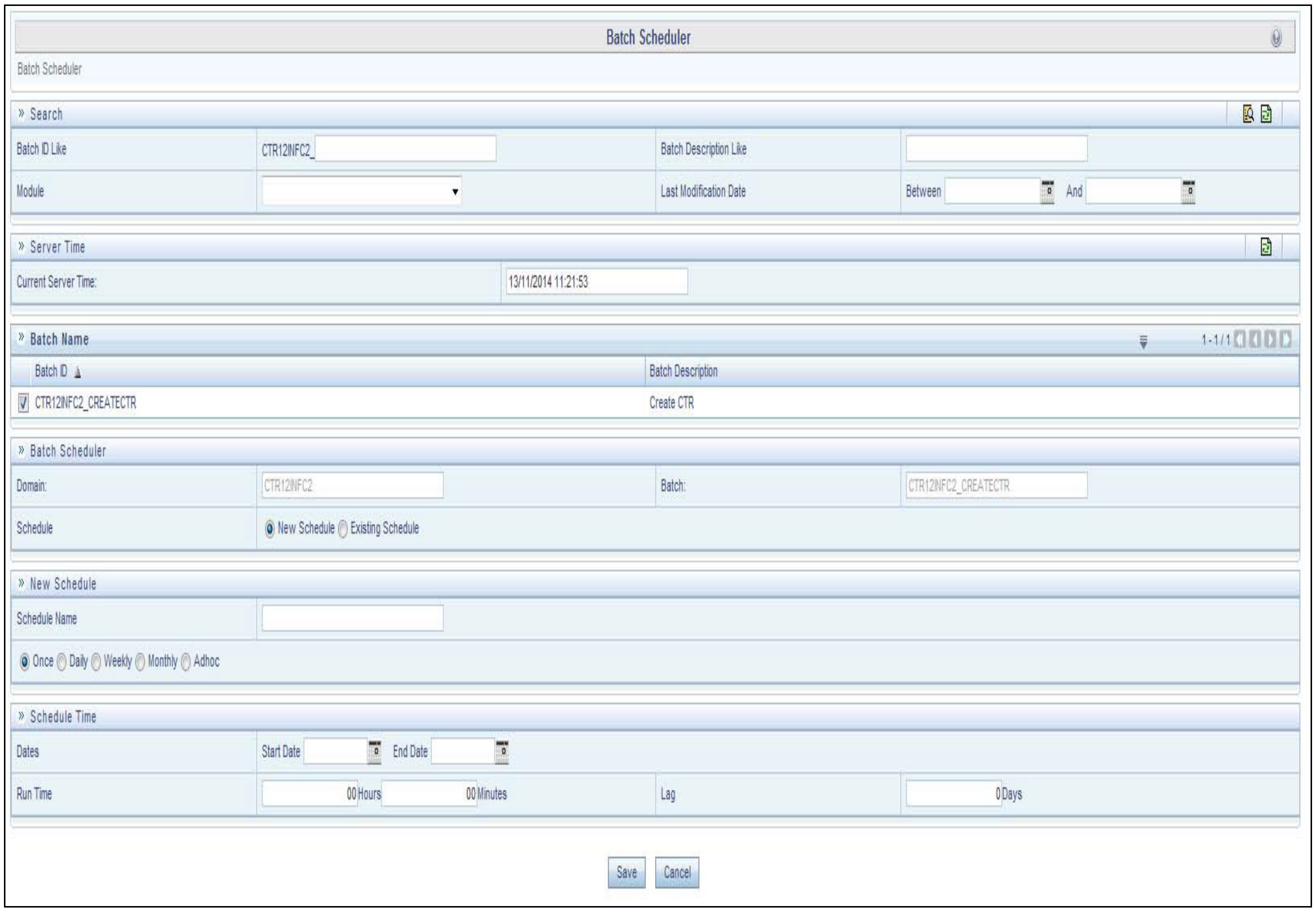
This section includes the following topics:

- Scheduling a Batch Once
- Scheduling a Daily Batch
- Scheduling a Weekly Batch
- Configuring a Monthly Batch

Scheduling a Batch Once

To schedule a batch that you want to run only once, follow these steps:

1. Login to Oracle Financial Services Analytical Applications Infrastructure as a CTR Administrator user.
2. Expand **Operations** from the LHS menu.
3. Click **Batch Scheduler**. The Batch Scheduler page is displayed.



Batch Scheduler

Batch Scheduler

» Search

Batch ID Like	CTR12INFC2_	Batch Description Like	
Module		Last Modification Date	Between And

» Server Time

Current Server Time:	13/11/2014 11:21:53

» Batch Name 1-1/1

Batch ID	Batch Description
✓ CTR12INFC2_CREATECTR	Create CTR

» Batch Scheduler

Domain:	CTR12INFC2	Batch:	CTR12INFC2_CREATECTR
Schedule	[✓ New Schedule] Existing Schedule		

» New Schedule

Schedule Name	

[✓ Once] Daily Weekly Monthly Adhoc

» Schedule Time

Dates	Start Date End Date		
Run Time	00 Hours 00 Minutes	Lag	0 Days

Figure 40. Batch Scheduler Page

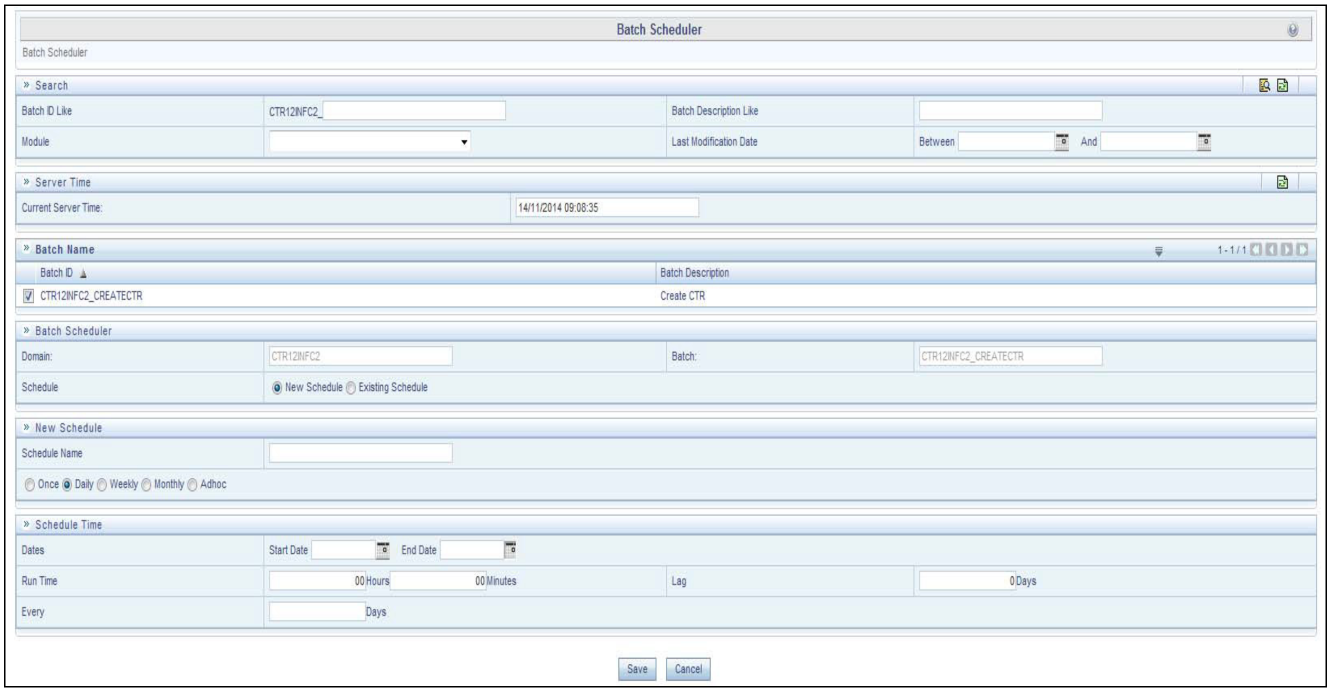
4. Select a batch that you want to schedule from the list of available batches. The Batch Scheduler section expands and displays additional options.
5. Select **New Schedule** radio button.
6. Set the frequency of the new schedule as Once by selecting the radio button.
7. Enter the schedule time of the batch by specifying the Start Date and the Run Time.
8. Click **Save**.

Scheduling a Daily Batch

To schedule a batch that you want to run daily, follow these steps:

1. Navigate to the Batch Scheduler page.
2. Select a batch that you want to schedule from the list of available batches. The Batch Scheduler section expands and displays additional options.
3. Click **New Schedule**.
4. Set the frequency of the new schedule as **Daily**.

5. Enter the schedule time of the batch by specifying the **Dates, Run Time,** and **Every field** information.



Batch Scheduler

Batch Scheduler

» Search

Batch ID Like	CTR12INFC2_	Batch Description Like	
Module		Last Modification Date	Between And

» Server Time

Current Server Time: 14/11/2014 09:08:35

» Batch Name

1 - 1 / 1

Batch ID	Batch Description
✓ CTR12INFC2_CREATECTR	Create CTR

» Batch Scheduler

Domain:	CTR12INFC2	Batch:	CTR12INFC2_CREATECTR
Schedule	[✓ New Schedule] Existing Schedule		

» New Schedule

Schedule Name

Once [✓ Daily] Weekly Monthly Adhoc

» Schedule Time

Dates	Start Date End Date		
Run Time	00 Hours 00 Minutes	Lag	0 Days
Every	Days		

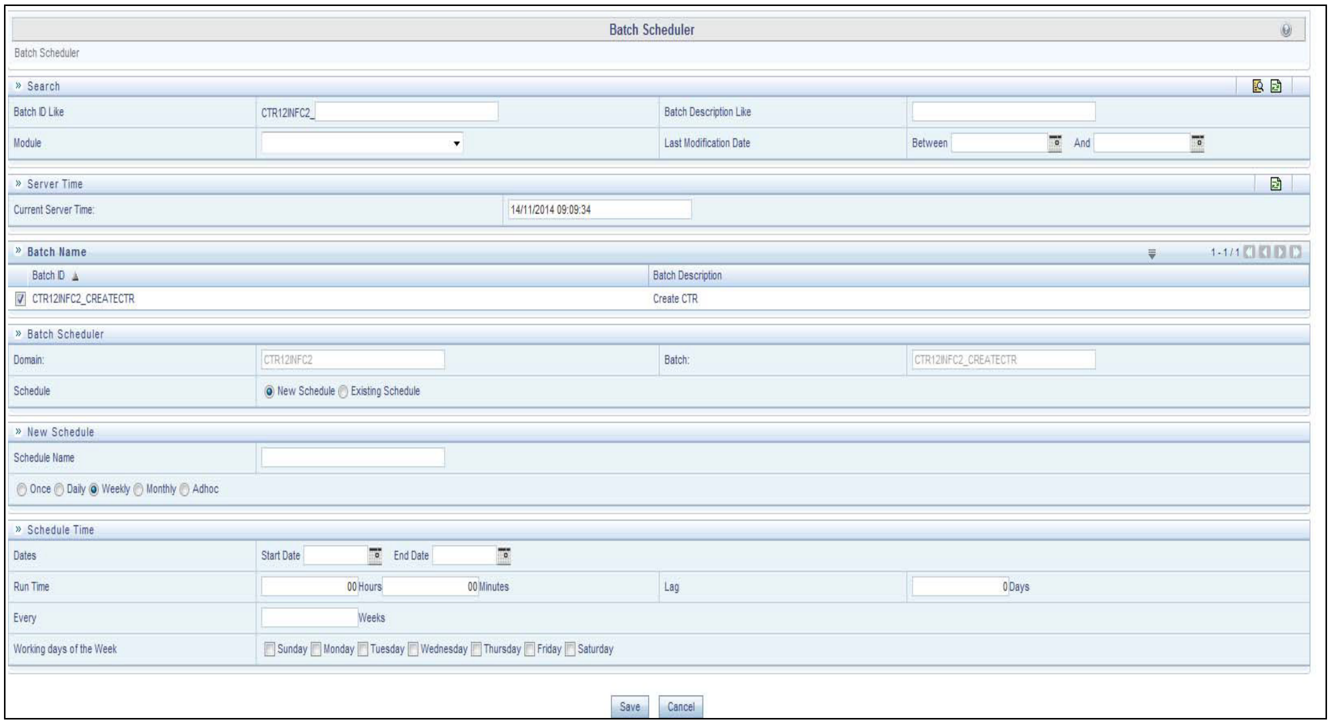
Figure 41. Scheduling a Daily Batch

6. Click **Save**.

Scheduling a Weekly Batch

To schedule a batch that you want to run weekly, follow these steps:

1. Navigate to the Batch Scheduler page.
2. Select a batch that you want to schedule from the list of available batches. The Batch Scheduler section expands and displays additional options.
3. Click **New Schedule**.
4. Set the frequency of the new schedule as **Weekly**.
5. Enter the schedule time of the batch by specifying the Dates, and other information such as **Run Time, Every,** and **Working days of the Week**.



Search

Batch ID Like	CTR12INFC2_	Batch Description Like	
Module		Last Modification Date	Between ___ And ___

Server Time

Current Server Time:	14/11/2014 09:09:34

Batch Name 1 - 1 / 1

	Batch ID	Batch Description
✓	CTR12INFC2_CREATECTR	Create CTR

Batch Scheduler

Domain:	CTR12INFC2	Batch:	CTR12INFC2_CREATECTR
Schedule	[✓ New Schedule] Existing Schedule		

New Schedule

Schedule Name	

Once Daily [✓ Weekly] Monthly Adhoc

Schedule Time

Dates	Start Date ___ End Date ___		
Run Time	00 Hours 00 Minutes	Lag	0 Days
Every	___ Weeks		
Working days of the Week	☐ Sunday ☐ Monday ☐ Tuesday ☐ Wednesday ☐ Thursday ☐ Friday ☐ Saturday		

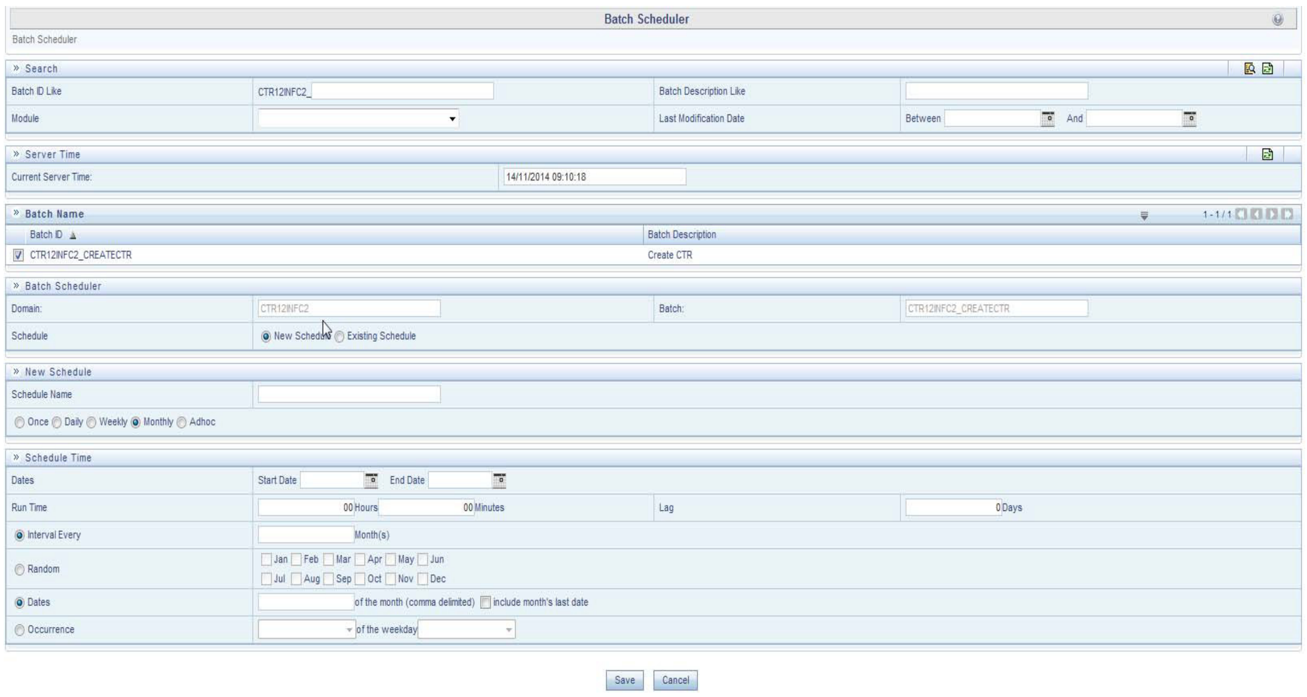
Figure 42. Scheduling a Weekly Batch

6. Click **Save**.

Configuring a Monthly Batch

To schedule a batch that you want to run monthly, follow these steps:

1. Navigate to the Batch Scheduler page.
2. Select a batch that you want to schedule from the list of available batches. The Batch Scheduler section expands and displays additional options.
3. Click **New Schedule**.
4. Set the frequency of the new schedule as **Monthly**.
5. Enter the schedule time of the batch by specifying the **Dates**, and **Run Time** information.



Batch Scheduler

Batch Scheduler

» Search

Batch ID Like	CTR12INFC2_	Batch Description Like	
Module		Last Modification Date	Between And

» Server Time

Current Server Time:	14/11/2014 09:10:18

» Batch Name

	Batch ID	Batch Description
✓	CTR12INFC2_CREATECTR	Create CTR

» Batch Scheduler

Domain:	CTR12INFC2	Batch:	CTR12INFC2_CREATECTR
Schedule	[✓ New Schedule] Existing Schedule		

» New Schedule

Schedule Name	

Once Daily Weekly [✓ Monthly] Adhoc

» Schedule Time

Dates	Start Date End Date		
Run Time	00 Hours 00 Minutes	Lag	0 Days
[✓ Interval Every]	Month(s)		
Random	Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec		
[✓ Dates]	of the month (comma delimited) include month's last date		
Occurrence	of the weekday		

Save Cancel

Figure 43. Configuring a Monthly Batch

6. Click **Save**.

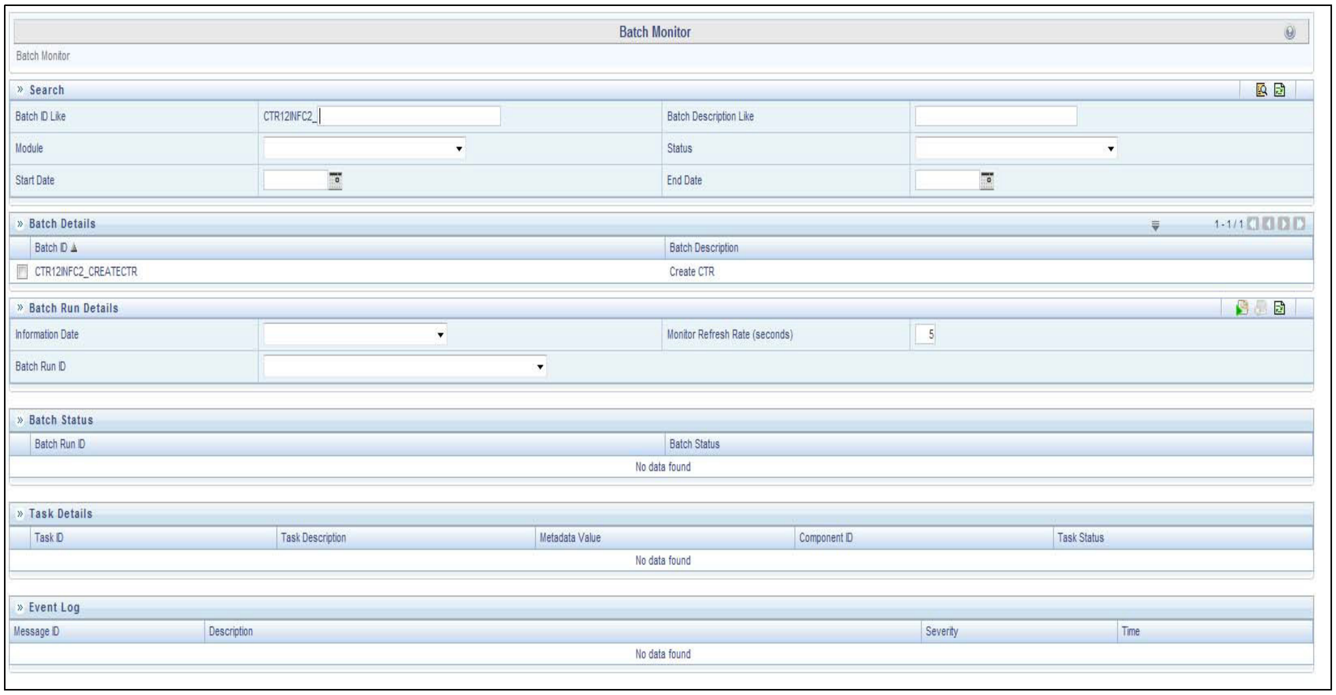
Monitoring a Batch After Execution

Monitoring a batch helps you track the status of execution of an individual task that was included in the batch. Through monitoring, you can also track the batch status, which in turn helps in debugging.

To monitor a batch after it is executed, follow these steps:

1. Login to Oracle Financial Services Analytical Applications Infrastructure as a CTR Administrator user.
2. Expand **Operations** from the LHS menu.

3. Click Batch Monitor. The Batch Monitor page is displayed.



Batch Monitor

Batch Monitor

» Search

Batch ID Like	CTR12INFC2_	Batch Description Like	
Module		Status	
Start Date		End Date	

» Batch Details 1 - 1 / 1

Batch ID ▲	Batch Description
CTR12INFC2_CREATECTR	Create CTR

» Batch Run Details

Information Date		Monitor Refresh Rate (seconds)	5
Batch Run ID			

» Batch Status

Batch Run ID	Batch Status
No data found	

» Task Details

Task ID	Task Description	Metadata Value	Component ID	Task Status
No data found				

» Event Log

Message ID	Description	Severity	Time
No data found			

Figure 44. Batch Monitor Page

4. Select a batch from the Batch Details lists that you want to monitor.
5. From the Batch Run Details section, select an Information Date and the Batch Run ID from the drop-down list.
6. Click to start the monitoring.
7. The execution details namely, Batch Status, Task Details, and Event Log details are displayed.

Cancelling a Batch after Execution

Cancellation of a batch cancels a current batch execution.

Note: This is not recommended and should be done only when the batch was fired accidentally or when a particular batch is taking too long time to execute.

To cancel a batch after it is executed, follow these steps:

1. Login to Oracle Financial Services Analytical Applications Infrastructure page as a CTR Administrator user.
2. Expand Operations from the LHS menu.
3. Click Batch Cancellation. The Batch Cancellation page is displayed.

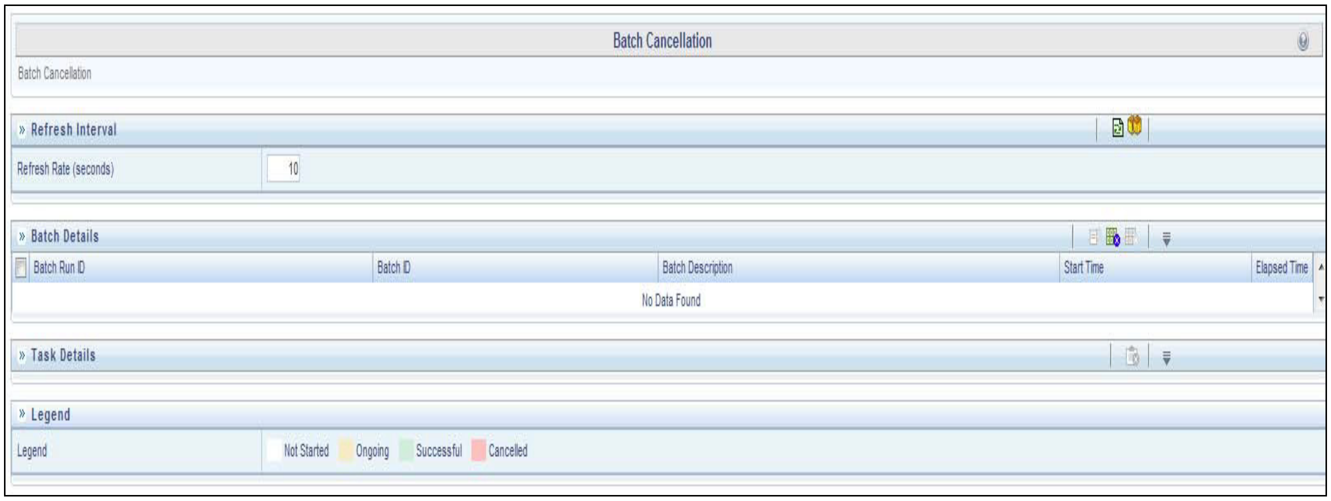


Figure 45. Batch Cancellation Page

4. Under the Batch Details section, select the batch whose execution you want to cancel.
5. Click **Cancel Batch**.

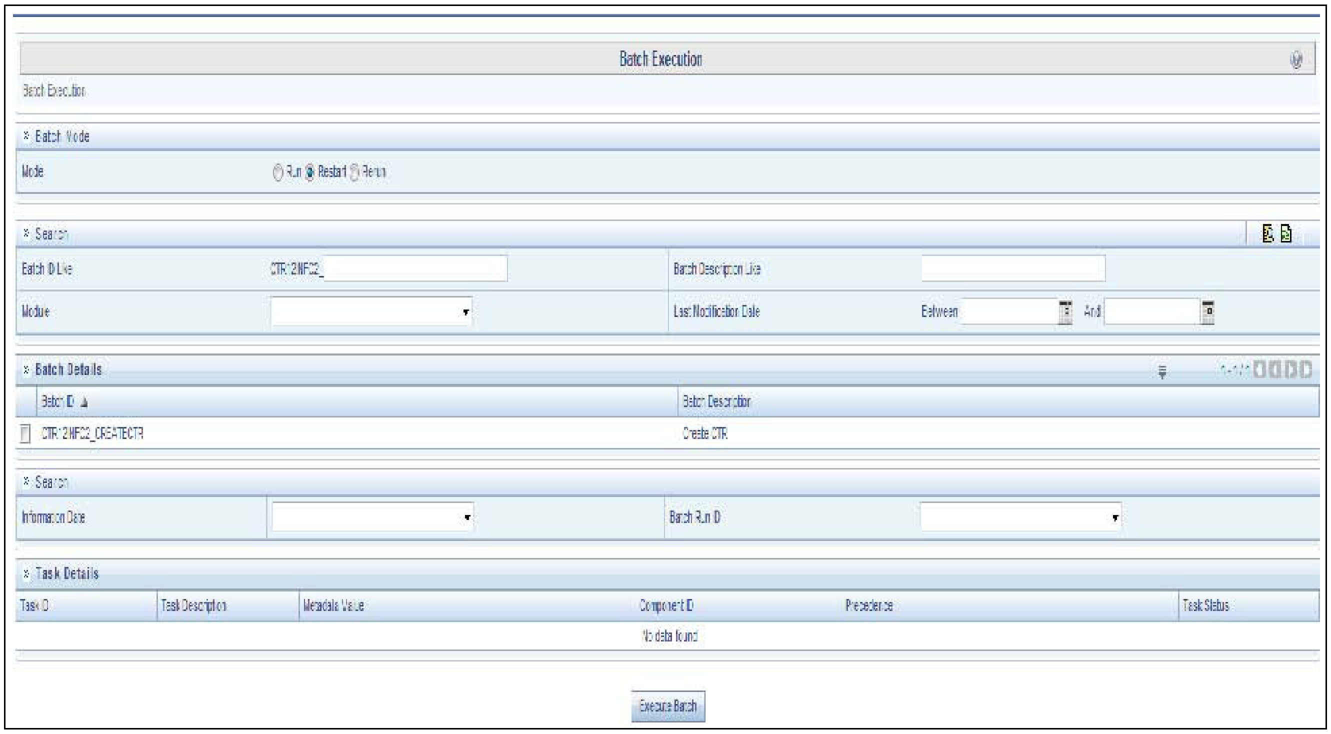
Re-starting a Batch

You can restart a batch execution when a batch has failed in execution. When you restart a batch, it starts from the task at which it had failed. This happens when the failed task issue is debugged and resolved.

Note: It is recommended that you debug and resolve a failed task before restarting the batch execution.

To restart a batch execution, follow these steps:

1. Login to Oracle Financial Services Analytical Applications Infrastructure as a CTR Administrator user.
2. Expand **Operations** from the LHS menu.
3. Click **Batch Execution.** The Batch Execution page is displayed.
4. Select the **Restart** radio button option from the Batch Mode section.


Batch Execution

Batch Execution

Batch Mode

Mode	Run [✓ Restart] Rerun

Search

Batch ID Like	CTR12NFC2_	Batch Description Like	
Module		Last Modification Date	Between And

Batch Details

Batch ID	Batch Description
CTR12NFC2_CREATECTR	Create CTR

Search

Information Date		Batch Run ID	

Task Details

Task ID	Task Description	Metadata Value	Component ID	Precedence	Task Status
			No data found		

Figure 46. Re-starting a Batch

5. Select the batch you want to restart from the Batch Details section.
6. Select the Information Date and Batch Run ID for the selected batch from the drop-down list.
7. Click **Execute Batch**.

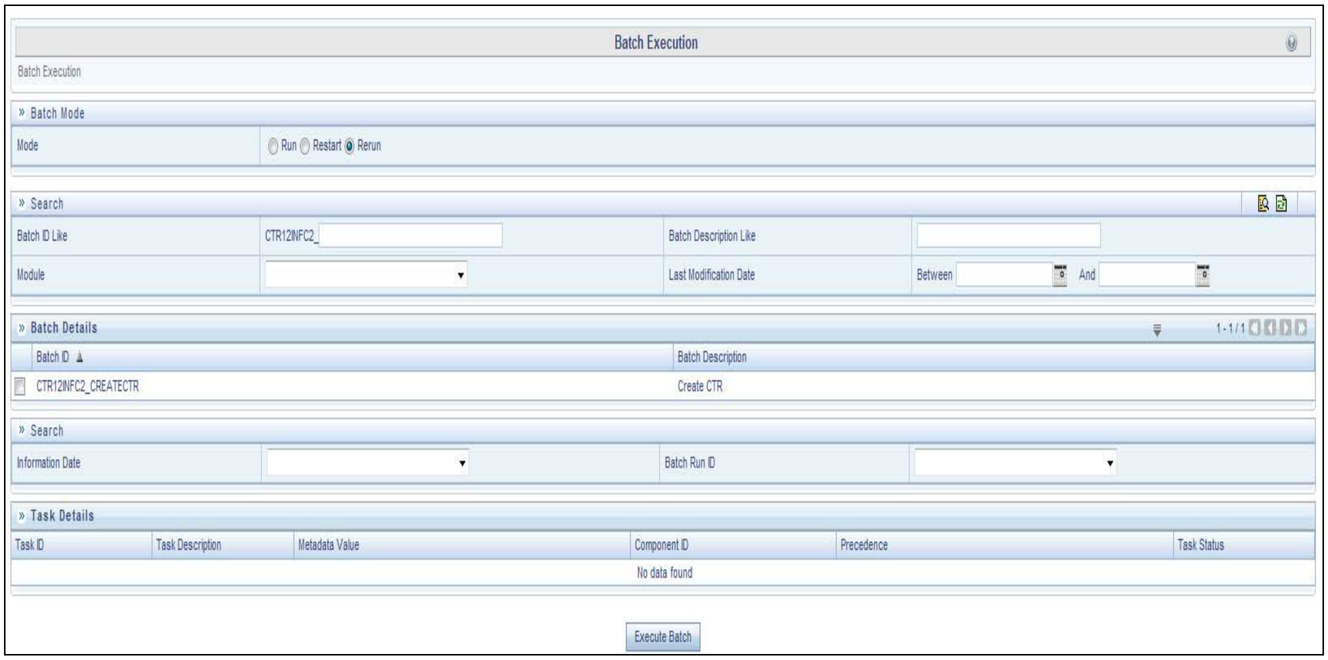
Re-running a Batch

You can rerun a batch execution when you want all the tasks from a successful batch execution to be executed again from the beginning. When a successfully executed batch is rerun, a different Batch Run ID is created for each instance for the same Information Date.

Note: Creating different Batch Run ID for each rerun of a batch is optional depending upon your firm's requirement.

To rerun a batch, follow these steps:

1. Login to Oracle Financial Services Analytical Applications Infrastructure as a CTR Administrator user.
2. Expand **Operations** from the LHS menu.
3. Click **Batch Execution**. The Batch Execution page is displayed.
4. Select the **Rerun** radio button from the Batch Mode section



Batch Execution

Batch Execution

» Batch Mode

Mode: Run Restart [✓ Rerun]

» Search

Batch ID Like	CTR12INFC2_	Batch Description Like	
Module		Last Modification Date	Between And

» Batch Details 1-1/1

Batch ID	Batch Description
CTR12INFC2_CREATECTR	Create CTR

» Search

Information Date		Batch Run ID	

» Task Details

Task ID	Task Description	Metadata Value	Component ID	Precedence	Task Status
			No data found		

Execute Batch

Figure 47. Re-running a Batch

5. Select the batch you want to rerun from the Batch Details section.
6. Select the Information Date and Batch Run ID for the selected batch from the drop-down list.
7. Click **Execute Batch**.

Batch Tasks

The tasks used in CTR Batch Process are listed in the table.

Table 48. Batch tasks description

Task Name	Description
Task 1	Creation of CTR
Task 2	Creation of CTR Notification
Task 3	Creation of Exemptions
Task 4	Logging Monetary Instrument Transactions

APPENDIX A Logging

This appendix describes the mechanism that Oracle Financial Services Behavior Detection Platform uses when logging system messages.

- About System Log Messages
- Message Template Repository
- Logging Levels
- Logging Message Libraries
- Logging Configuration File

About System Log Messages

The Common Logging component provides a centralized mechanism for logging Behavior Detection messages, in which the system places all log messages in a single message library file.

In the event that a log file becomes very large (one gigabyte or more), the system creates a new log file. The naming convention is to add *.x* to the log file's name (for example, `mantas.log`, `mantas.log.1`, `mantas.log.2`, so forth).

Note: The log file size is a configurable property; section *Log File Sizes,* on page 138, provides instructions. The default value for this property is 10 MB. The maximum file size should not exceed two gigabytes (2000000000 bytes).

Message Template Repository

The message template repository resides in a flat text file and contains messages in the format `<message ID 1> <message text>`. The following is an example of a message repository's contents:

```
111 Dataset ID {0} is invalid
112 Run ID {0} running Pattern {1} failed
113 Checkpoint false, deleting match
```

111, 112, and 113 represent message IDs; whitespace and message text follow. The {0}s and {1}s represent placeholders for code variable values.

Each subsystem has its own repository.

The naming convention for each message library file is `mantas_<subsystem>_message_lib_<language-code>.dat`, where `<subsystem>` is the name of the subsystem and `<language-code>` is the two-character Java (ISO 639) language code. For example, the English version of the Algorithms message library is `mantas_algorithms_message_lib_en.dat`.

The `log.message.library` property that the subsystem's base `install.cfg` file contains specifies the full path to a subsystem's message library file.

Logging Levels

Table 49 outlines the logging levels that the Common Logging component supports.

Table 49. Logging Levels

Severity (Log Level)	Usage
Fatal	Irrecoverable program, process, and thread errors that cause the application to terminate.
Warning	Recoverable errors that may still enable the application to continue running but should be investigated (for example, failed user sessions or missing data fields).
Notice (default)	High-level, informational messaging that highlights progress of an application (for example, startup and shutdown of a process or session, or user login and logout).
Diagnostic	Fine-grained diagnostic errors—used for viewing processing status, performance statistics, SQL statements, etc.
Trace	Diagnostic errors—use only for debugging purposes as this level enables all logging levels and may impact performance.

The configuration file specifies enabling of priorities in a hierarchical fashion. That is, if Diagnostic is active, the system enables the Notice, Warning, and Fatal levels.

Logging Message Libraries

Some Behavior Detection subsystems produce log output files in default locations. The following sections describe these subsystems.

Administration Tools

The following file is the message library for the Administration Tools application:

```
<FIC_HOME>/AM/admin_tools/WEB-INF/classes/conf/mantas_cfg/etc/mantas_admin_tools_m
essage_lib_en.dat
```

All messages numbers that this log contains must be within the range of 50,000 - 89,999.

Database

The following file is the message library for the Database:

```
<INSTALL_DIR>/database/db_tools/mantas_cfg/etc/
mantas_database_message_lib_en.dat
```

All messages numbers that this file contains must be within the range of 250,000 - 289,999.

Database objects logs

DB objects logs used in the application are maintained in the table `KDD_LOGS_MSGS`. An entry in this table represents the timestamp, stage, error code and module.

Logging Configuration File

You can configure common logging through the following files depending on the subsystem you want to modify:

- Database:
  `<INSTALL_DIR>/database/db_tools/mantas_cfg/install.cfg`
- Behavior Detection: `<INSTALL_DIR>/behavior_detection/algorithms/MTS/mantas_cfg/install.cfg`

The configuration file specifies enabling of priorities in a hierarchical fashion. For example, if Diagnostic priority is enabled, Notice, Warning, and Fatal are also enabled, but Trace is not.

In the configuration file, you can specify the following:

- Locations of recorded log messages
- Logging to the console, files, UNIX syslog, e-mail addresses, and the Microsoft Windows Event Viewer
- Routing based on severity and/or category
- Message library location
- Maximum log file size

Sample Configuration File

The following is a sample logging configuration file. Make special note of the comments inthe following sample as they contain constraints that relate to properties and logging.

```
# Specify which priorities are enabled in a hierarchical fashion, i.e., if
# DIAGNOSTIC priority is enabled, NOTICE, WARN, and FATAL are also enabled,
# but TRACE is not.
# Uncomment the desired log level to turn on appropriate level(s).
# Note, DIAGNOSTIC logging is used to log database statements and will slow
# down performance. Only turn on if you need to see the SQL statements being
# executed.
# TRACE logging is used for debugging during development. Also only turn on
# TRACE if needed.
#log.fatal=true
#log.warning=true
log.notice=true
#log.diagnostic=true
#log.trace=true

# Specify whether logging for a particular level should be performed
# synchronously or asynchronously.
log.fatal.synchronous=false
log.warning.synchronous=false
log.notice.synchronous=false
log.diagnostic.synchronous=false
log.trace.synchronous=true

# Specify the format of the log output. Can be modified according to the format
# specifications at:
# http://logging.apache.org/log4j/docs/api/org/apache/log4j/PatternLayout.html
# NOTE: Because of the nature of asynchronous logging, detailed information
# (class name, line number, etc.) cannot be obtained when logging
# asynchronously. Therefore, if this information is desired (i.e. specified
# below), the above synchronous properties must be set accordingly (for the
# levels for which this detailed information is desired). Also note that this
# type of detailed information can only be obtained for Java code.
log.format=%d [%t] %p %m%n

# Specify the full path and file name of the message library.
log.message.library=@WORKFLOW_LOG_MESSAGE_LIB_FILE@

# Specify the full path to the categories.cfg file
log.categories.file.path=@WORKFLOW_LOG_CATEGORY_PATH@

# Multiple locations can be listed for each property using a comma delimiter.

log.category.TEST_CATEGORY.location=console, mantaslog
log.category.TEST_CATEGORY_2.location=console, /users/jsmith/logs/mylog.log
```

(Continued on next page)

(Continued from previous page)

```
# Specify where messages of a specific severity and category should be logged to.
# The valid values are the same number as for category.
# Multiple locations can be listed for each property using a comma delimiter.
# If an entry for a severity is not listed here, the message is logged to
# the location specified for the category number by the above property, and if that does
not exist, it is logged to the default location configured below.

log.TEST_CATEGORY.warning.location=syslog
log.TEST_CATEGORY.fatal.location=user@domain.com
log.TEST_CATEGORY_2.warning.location=syslog

# # Specify the full path to the external log4j configuration file
log4j.config.file=@WORKFLOW_LOG4J_CONFIG_FILE@

# Specify where a message should get logged for a category for which there is
# no location property listed above.
# This is also the logging location of the default Oracle Financial Services category
unless
# otherwise specified above.
# Note that if this property is not specified, logging will go to the console.
log.default.location=

# Specify the location (directory path) of the mantaslog, if the mantaslog
# was chosen as the log output location anywhere above.
# Logging will go to the console if mantaslog was selected and this property is
# not given a value.
log.mantaslog.location=

# Specify the hostname of syslog if syslog was chosen as the log output location
# anywhere above.
# Logging will go to the console if syslog was selected and this property is
# not given a value.
log.syslog.hostname=

# Specify the hostname of the SMTP server if an e-mail address was chosen as
# the log output location anywhere above.
# Logging will go to the console if an e-mail address was selected and this
# property is not given a value.
log.smtp.hostname=

# Specify the maxfile size of a logfile before the log messages get rolled to
# a new file (measured in bytes).
# If this property is not specified, the default of 10 MB will be used.
```

Figure 48. Sample Logging Configuration File

Logging Location Property Values

The `log.category.<CATEGORY_NAME>.location` property enables you to specify the location of a message for a specific category. If you do not specify a location value, the system logs messages in a default location.

Table 50 identifies the valid values for this property.

Table 50. Logging Location Property Values

Property value	Log location
`console`	Records the logs to the `system.out` or `system.err` file.
`syslog`	Records the logs to a remote UNIX syslog daemon. This is the default location.
`eventviewer`	Records the logs to the Event Log system.
mantaslog	Indicates the format of the mantaslog filename as job<job #>-datetimestamp (if running the algorithms). For other subsystems, the format is mantaslog-datetimestamp. The file resides at the location that the `log.mantaslog.location` property specifies in the appropriate `install.cfg` file. If this property is unspecified, the system outputs logs to the console.
`<path>/<filename>`	Records the logs to a file with the filename <filename>, which resides at <path>. For example, `log.message.library=/user/jsmith/message/messages.dat`
`<name@address>`	Records the logs in a message to the e-mail address indicated by `<name@address>`.

Note that category names (that is, property values) cannot contain the following reserved words: fatal, warning, notice, diagnostic, trace, category, or location. You can configure multiple locations for each property using a comma delimiter.

For example:

```
log.category.TEST_CATEGORY.location=console, mantaslog
log.category.TEST_CATEGORY_2.location=console, /users/jsmith/logs/mylog.log
```

Log File Sizes

If an Currency Transaction Reporting client chooses to record log messages to files, those log files may become very large over the course of time, or depending on the types of logging enabled. If this occurs, the system rolls files larger than 10 MB (if `log.max.size` property is not specified) over to another log file and adds a number incrementally to the log file name. For example, if your log file name is `mantas.log`, additional log files appear as `mantas.log.1`, `mantas.log.2`, so forth.

Note: The maximum value for the `log.max.size` property can be 2000000000.

Configurable Logging Properties

Table 51 identifies the configurable properties for logging in an Oracle client's environment.

Table 51. Configurable Parameters for Common Logging

Property	Sample Value	Description
`log.format`	%d [%t] %p %m%n	Identifies the log formatting string. See Apache Software's *Short Introduction to log4j* guide (http://logging.apache.org/log4j/docs/ manual.html) for more details about the log message format.
`log.message.library`	To be specified at installation.	Identifies the full path and filename of the message library.
`log.max.size`	2000000000	Determines the maximum size (in bytes) of a log file before the system creates a new log file. For more information (See *Log File Sizes,* on page 138, for more information).
`log.category.<catgory_name>. location`		Contains routing information for message libraries for this category. For more information (See *Logging Location Property Values,* on page 138, for more information).
`log.categories.file.path`	To be specified at installation.	Identifies the full path to the `categories.cfg` file.
`log.<category_name>. <severity>.location`		Contains routing information for message libraries with the given severity for the given category. For more information (See *Logging Location Property Values,* on page 138, for more information).
`log4j.config.file`	To be specified at installation.	Specifies the full path to the external log4j configuration file.
`log.default.location`		Contains routing information for message libraries for this category for which there is no location previously specified.
`log.mantaslog.location`		Contains routing information for message libraries for this category for which there is no location previously specified.
`log.syslog.location`		Contains routing information for message libraries for this category for which there is no location previously specified.
`log.smtp.hostname`		Identifies the hostname of the SMTP server if e-mail address is specified as log output.
`log.fatal`	true	Indicates that fatal logging is enabled; *false* indicates that fatal logging is not enabled.
`log.fatal.synchronous`	false	Indicates that fatal logging is enabled; *false* indicates that fatal logging is not enabled.
`log.warning`	true	Indicates enabling of warning logging; *false* indicates that warning logging is not enabled.

Table 51. Configurable Parameters for Common Logging (Continued)

Property	Sample Value	Description
`log.warning.synchronous`	false	Indicates enabling of warning logging; *false* indicates that warning logging is not enabled.
`log.notice`	true	Indicates enabling of notice logging; *false* indicates that notice logging is not enabled.
`log.notice.synchronous`	false	Indicates enabling of notice logging; *false* indicates that notice logging is not enabled.
`log.diagnostic`	false	Indicates that diagnostic logging is not enabled; *true* indicates enabling of diagnostic logging.
`log.diagnostic.synchronous`	false	Indicates that diagnostic logging is not enabled; *true* indicates that diagnostic logging is enabled.
`log.trace`	false	Indicates that trace logging is not enabled; *true* indicates enabling of trace logging.
`log.trace.synchronous`	true	Indicates that trace logging is not enabled; *true* indicates enabling of trace logging.
`log.syslog.hostname`	hostname	Indicates the host name of syslog for messages sent to syslog.
`log.smtp.hostname`	hostname	Indicates the host name of the SMTP server for messages that processing sends to an e-mail address.
`log.time.zone`	EST	Indicates the time zone that is used when logging messages.

Ingestion uses common logging by assigning every component (for example, FDT or MDT) a category. You can configure the destination of log messages for each component which Table 50 describes. The default logging behavior is to send log messages to the component's designated log file in the `date` subdirectory representing the current processing date under the `logs` directory. This behavior can be turned off by setting the `Log@UseDefaultLog` attribute in `DataIngest.xml` to false. If this attribute is true, the system continues to send messages to the designated log files in addition to any behavior that the common logging configuration file specifies.

Monitoring Log Files

When using a tool to monitor a log file, use the message ID to search for a particular log message instead of text within the message itself. Under normal circumstances, the message IDs are not subject to change between Currency Transaction Reporting releases, but the text of the message can change. If a message ID does change, you can See the appropriate `readme.txt` file for information about updated IDs.