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

Release 1.1
September 2012

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

Release 1.1
September 2012

Document Control Number: 9MN16-0002
Document Number: AG-12-CTR-0002-6.1-1.1-01

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

Document Number: AG-12-CTR-0002-6.1-1.1-01
First Edition (September 2012)

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 1.1 Administration Guide*.

Table 1. Revision History

Date	Edition	Description
September 2012	First edition.	The first release of *Oracle Financial Services Currency Transaction Reporting 1.1 Administration Guide.*

Contents

List of Tables

List of Figures

About this Guide

This guide explains the concept behind the Oracle 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 user 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 the Oracle Financial Services Installers and System Administrators. Their roles and responsibilities, as they operate within the Oracle Financial Services, 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 Financial Services 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, *The Oracle Financial Services Behavior Detection Platform,* provides a brief overview of the Oracle Financial Services 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 Oracle Financial Services jobs.
- Chapter 3, *Security Configuration,* covers the required day-to-day operations and maintenance of Oracle Financial Services 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 in Informatica, after the Oracle Financial Services data ingestion process completes.
- Chapter 6, *Batch Processing Utilities,* explains how to customize Oracle Financial Services 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.
- 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, refer to

- *Oracle Financial Services Behavior Detection Platform Installation Guide - Stage 1*
- *Oracle Financial Services Currency Transaction Reporting Installation Guide - Stage 3*
- *Oracle Financial Services Currency Transaction Reporting Data Interface Specification Guide, Release 1.1*
- *Oracle Financial Services Currency Transaction Reporting Configuration Guide, Release 1.1*
- *Oracle Financial Services Currency Transaction Reporting User Guide, Release 1.1*
- *Oracle Financial Services Currency Transaction Reporting Release Notes, Release 1.1*
- *Oracle Financial Services Analytical Applications Infrastructure Installation and Configuration, Release 7.3*
- *Oracle Financial Services Analytical Applications Infrastructure User Manual, Release 7.3*

For installation and configuration information about Sun Java System, BEA, and Apache software, refer to 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

The Oracle Financial Services Behavior Detection Platform

This chapter provides a brief overview of the Oracle Financial Services Behavior Detection Platform in terms of its 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, and Oracle. Refer to the *Oracle Financial Services Behavior Detection Platform Stage 1 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 Oracle Financial Services architecture:

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

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.

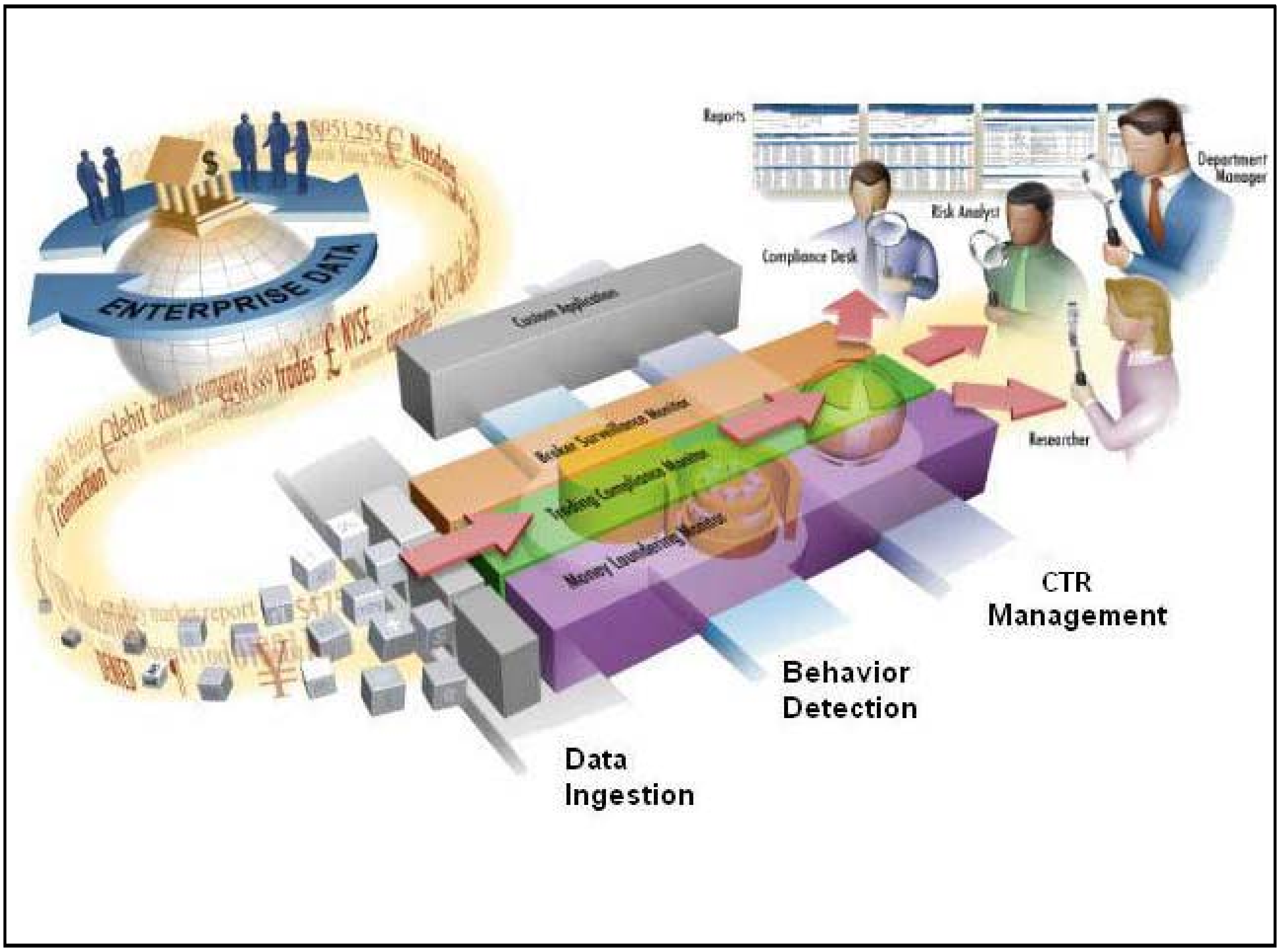


Figure 1. Oracle Financial Services Architecture - Overview

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.

The Oracle Financial Services solution has two tiers:

- **Oracle Financial Services Behavior Detection Platform** defines a foundation for building Oracle Financial Services solution sets. It provides core data mining services, frameworks, and tools. Oracle Financial Services also includes interface packages that abstract non-standard or proprietary commercial off-the-shelf (COTS) products. Deployment of multiple Oracle

Financial Services solution sets can occur on a single Oracle Financial Services 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

Oracle Financial Services is composed of below subsystems, they are as follows:

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

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

Table 3. Oracle Financial Services Subsystems and their Components

Common Name	Directory Name	Contents
Data Ingestion	`ingestion_manager`	Java components, scripts, and stored procedures
Database Tools	`database/db_tools`	For DB tools directory
Detection Algorithms	`algorithms`	C++ behavior detection algorithms
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 an Behavior Detection architecture deployment is illustrated in Figure 1. 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 Financial Services 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.

Behavior Detection 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..

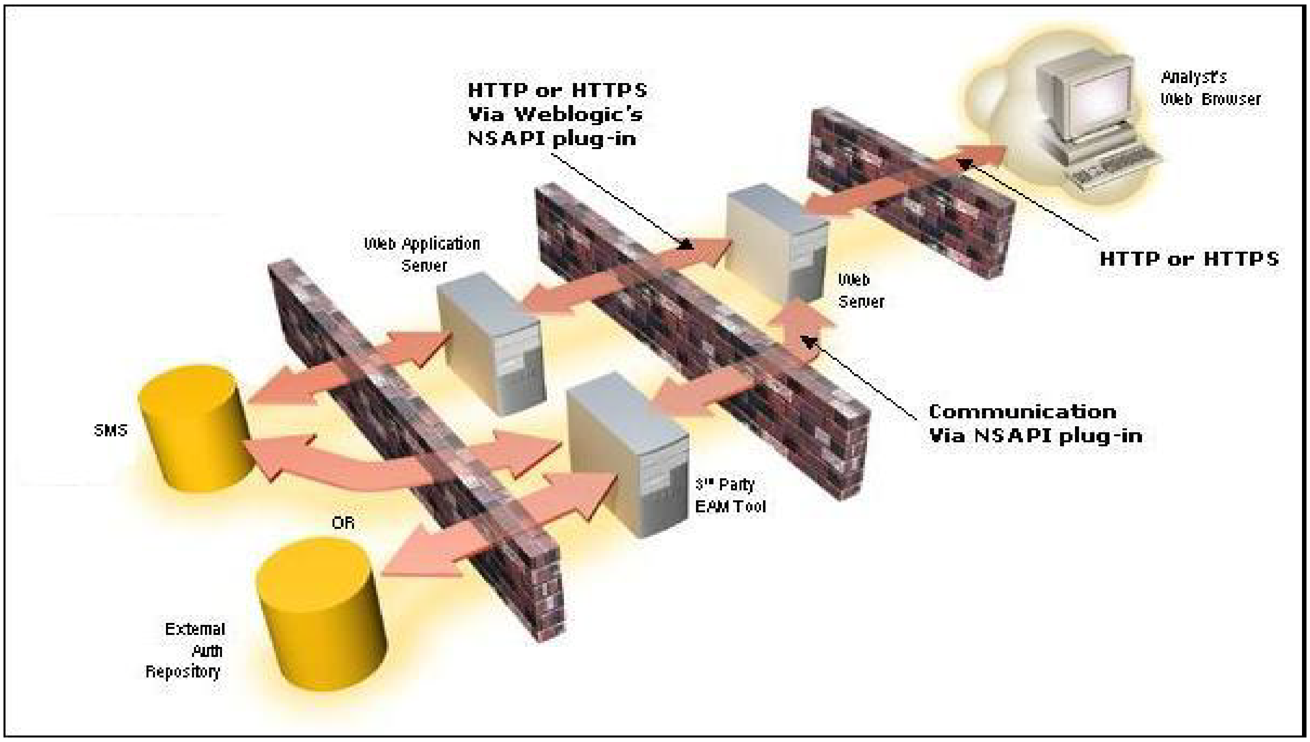


Figure 2. Oracle Financial Services Architecture—Security View

About Oracle Financial Services Operations

As the Oracle Financial Services administrator, you coordinate the overall operations of Oracle Financial Services: Data Ingestion, Behavior Detection, and Post-Processing.

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

As Oracle Financial Services Architecture—Oracle Financial Services Processing illustrates, each cycle of Oracle Financial Services 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 an Oracle Financial Services batch process (refer to Chapter 6, *Batch Processing Utilities*, on page 83, for more information).

Data Ingestion

The Oracle Financial Services Ingestion Manager controls the data ingestion process. The *Oracle Financial Services Currency Transaction Reporting Data Interface Specification Guide, Release 1.1* contains a definition of each solution set.

The Ingestion Manager supports files and messages for the ingestion of data. Data ingestion involves receiving source data from an external data source in one of these forms. The Ingestion Manager validates this data against the DIS and populates the Oracle Financial Services database with the results (refer to Chapter 4, *Data Ingestion*, on page 37, for more information).

Behavior Detection

During Behavior Detection, Oracle Financial Services 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.

Post-Processing

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

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. Refer to *To Run Match Alert Creator,* on page 76, for more information.
2. During batch execution, alerts are converted into CTR records. Refer to *Batch Execution of CTR,* on page 76, for more information.

End Batch

The system ends batch processing when processing of data from the Oracle Financial Services client is complete (Refer to section *Ending a Batch Process,* on page 114, 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

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.

Refer to *Starting the Dispatcher*, on page 12, and *Stopping the Dispatcher*, on page 13, 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 Financial Services 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 `dispatcher`s on one machine.
   - `JVM size`: Indicates the amount of memory to allocate to Java processing.

CAUTION: For 32-bit Linux configurations, Oracle Financial Services 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 the 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 the 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 Financial Services administrator can start, restart, copy, stop, monitor, and diagnose jobs.

The sections below cover 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 Financial Services administrator starts jobs by running the `start_mantas.sh` script.

To start a new job, follow the steps:

1. Create the new job and job description through an 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 (Refer to 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 the steps:

1. Create the new job and job description through an 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 (Refer to section *Starting Jobs*, on page 15, 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 Refer to 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 Financial Services 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 (Refer to section *Stopping the Dispatcher*, on page 13, 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 the steps:

1. Stop the `dispatcher` by following the procedure for Stopping the dispatcher (Refer to section *Stopping the Dispatcher*, on page 13, 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 Oracle Financial Services Behavior Detection user authentication and authorization. It also contains instructions for setting up user accounts in the Oracle Financial Services Behavior Detection database to access the Currency Transaction Reporting. This chapter focuses on the following topics:

- About the 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

About the Oracle Financial Services User Authentication

The primary way to access Oracle Financial Services Behavior Detection information is through a Web browser that accesses the Currency Transaction Reporting. Oracle Financial Services Behavior Detection offers SMS for authentication of web browser clients. Behavior Detection offers the following authentication mechanism for Web browser clients:

- Built-in Authentication System and Security Management System (SMS) on the Web Application server that authenticates users from a login Web page. (Refer *Understanding SMS* for more information).

Understanding SMS

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 user 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.

For accessing Oracle Financial Services Behavior Detection:

- Set of policies that associate 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: Refer to the *Oracle Financial Services Analytical Applications Infrastructure User Manual, Release 7.3* 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.

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

Note: For the above sections, refer to *Oracle Financial Services Analytical Applications Infrastructure User Manual, Release 7.3* 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, Refer to the below mentioned sections of the *Oracle Financial Services Analytical Applications Infrastructure User Manual, Release 7.3:*

- 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

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. Refer to *Oracle Financial Services Analytical Applications Infrastructure User Manual, Release 7.3* 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 Financial Services client to model client data along operational business lines and practices.
- **Jurisdiction(s):** Property that enables an Oracle Financial Services 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 Financial Services 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 would need to take Email action. The user ID should configured with valid email IDs while configuring the same 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 Financial Services client that the Ingestion Manager 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,
JRSDCN_DESC_TX) VALUES ('N', 'North', 'North', 'Northern US');
```

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 the Ingestion Manager 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. Hence, 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 the 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');

COMMIT;
```

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 using the below 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 `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 privilege of accessing CTR records that belong to the mapped business domains. This is done when an entry is made in the KDD_REVIEW_OWNER table (Refer to *Mapping Users To Access Control Metadata,* on page 28,). 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.
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

Table 16. KDD_ORG table Attributes

ORG_TYPE_CD	Identifier of the Oracle Financial Services 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 Financial Services 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 Financial Services client’s discretion.
CSTM_2_DT	Date field that is available for use at the Oracle Financial Services client’s discretion.
CSTM_3_DT	Date field that is available for use at the Oracle Financial Services client’s discretion.
CSTM_1_RL	Number field that is available for use at the Oracle Financial Services client’s discretion.
CSTM_2_RL	Number field that is available for use at the Oracle Financial Services client’s discretion.
CSTM_3_RL	Number field that is available for use at the Oracle Financial Services client’s discretion.
CSTM_1_TX	Text field that is available for use at the Oracle Financial Services client’s discretion.
CSTM_2_TX	Text field that is available for use at the Oracle Financial Services client’s discretion.
CSTM_3_TX	Text field that is available for use at the Oracle Financial Services client’s discretion.
CSTM_4_TX	Text field that is available for use at the Oracle Financial Services client’s discretion.
CSTM_5_TX	Text field that is available for use at the Oracle Financial Services 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	
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

Table 16. KDD_ORG table Attributes

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.
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 below 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 the below steps to set the user defined home page:

1. Click on the Home button from the LHS menu when you log into the OFSAAI for the first time.
2. An option to select the default screen for the user appears.
3. Select CTR from the drop-down option, and click on Save
4. The CTR home page is set as the landing page for the user.

CHAPTER 4

Data Ingestion

This chapter discusses the operation of the Oracle Financial Services Data Ingestion processor, Ingestion Manager, and subsystem components. Specifically, this chapter focuses on the following topics:

- Process Flow
- Data Ingestion Directory Structure
- Startup and Shutdown
- Data Rejection During Ingestion
- Data Ingestion Archiving

Process Flow

The Data Ingestion subsystem components receive and process data in a series of workflow steps that include extract or preprocess, transform, load, and post-load transformations.

Data Ingestion Directory Structure

The processes within each of the procedures refer to input and output directories within the Data Ingestion directory structure. Where not called out in this chapter, all Data Ingestion directories (for example, `/inbox` or `/config`) reside in `<INSTALL_DIR>/ingestion_manager`.

Also, processing datestamps many Data Ingestion directories and subdirectories so that they appear with a *YYYYMMDD* notation. The system provides this processing date to the `set_mantas_date.sh` shell script when starting the first batch for the day.

For detailed information about the Data Ingestion directory structure, refer to section *Data Ingestion Directory Structure*, on page 41, for more information.

Beginning Preprocessing and Loading

The system executes preprocessors using the `runDP.sh` script. The following sample command shows invoking of a preprocessor:

```
<INSTALL_DIR>/ingestion_manager/scripts/runDP.sh Account
```

Ingestion Manager processes data files in groups (in a specified order) from Oracle Financial Services client data in the `/inbox` directory.

Table 17 lists the data files by group.

Table 17. Data Files by Group

Group	Data Files	
1	AccountCustomerRole Organization Currency Transaction	
3	Account Customer OrganizationRelationship	
4	AccountToCustomer CustomerAddress CustomerEmailAddress CustomerPhone	BranchCTRConductor BranchCTRTransaction BranchCTRSummary

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 Manager 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 Financial Services 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. These dependencies are as follows:

Process Flow

The ingestion process flow is as follows:

1. Behavior Detection receives firm data in ASCII flat `.dat` files, which an Oracle Financial Services client's data extraction process places in the `/inbox` directory. This data can be:

    - Reference (for example, point-in-time customer and account data)
    - Transactional (for example, market and trading data)

    The preprocessor addresses only those files that match naming conventions that the DIS describes, and which have the date and batch name portions of the file names that match the current data processing date and batch.

    The Oracle Financial Services client need only supply those file types that the solution sets require.

2. Ingestion Manager executes preprocessors simultaneously (within hardware capacities). The preprocessors use XML configuration files in the `/config/datamaps` directory to verify that the format of the incoming Oracle Financial Services client data is correct and validate its content; specifically:

    - Error-checking of input data
    - Assigning sequence IDs to records

- Resolving cross-references to reference data
- Checking for missing records
- Flagging data for insertion or update

Preprocessors place output files in the directories that Table 18 lists.

Table 18. Preprocessing Output Directories

Directory Name	Description
`/inbox/<yyyymmdd>`	Backup of input files (for restart purposes, if necessary).
`/data/<business or market>/load`	• Data files for loading into the database as `<data type>_<yyyymmdd>_<batch name>_<N>.XDP`. • Load control files.
`/logs/<yyyymmdd>`	Preprocessing and load status, and error messages.
`/data/errors/<yyyymmdd>`	Records that failed validation. The file names are the same as those of the input files.
`/data/firm/transform`	TC trading data files that the FDT processes.

3. Simultaneous execution of `runDL.sh` scripts (within hardware capacities) loads each type of data into the FSDM. This script invokes a data loader to load a specified preprocessed data file into the database.

   For reference data (any file that has a load operation of *Overwrite,* which the DIS specifies), two options are available for loading data:

   - **Full Refresh:** Truncating of the entire table occurs before loading of data. This mode is intended for use when a client provides a complete set of records daily.
   - **Delta Mode:** Updating of existing data and insertion of new data occur. This mode is intended for use when a client provides only new or changed records daily.

   The `FullRefresh` parameter in `DataIngest.xml` controls the use of full refresh or delta mode. When this parameter is *true*, the system uses full refresh mode; when it is *false*, the system uses delta mode. Setting the default can be for either mode; overriding the default for individual file types is also possible, when needed.

   The following sample command illustrates execution of data loaders:

   ```
   <INSTALL_DIR>/ingestion_manager/scripts/runDL.sh Account
   ```

Guidelines for Duplicate Record Handling

The Ingestion Manager considers records as duplicates if the primary business key for multiple records are the same. The Ingestion Manager manages these records by performing either an insert or update of the database with the contents of the first duplicate record. The system inserts the record if a record is not currently in the database with the same business key. The record updates the existing database record if one exists with the same business key. The Ingestion Manager handles additional input records with the same business key by performing database updates. Therefore, the final version of the record reflects the values that the last duplicate record contains.

Data Ingestion Directory Structure

The Data Ingestion subsystem components and data are organized in subdirectories below the `ingestion_manager` root level. Table 19 provides details about each subdirectory.

Installation of the Data Ingestion subsystem is normally on a single server. When requiring high availability or improved performance, however, installation on two or more servers is common. Oracle Financial Services recommends installation of the subsystem on a server that has sufficient, direct-connected RAID disk storage for both the product and ingested data. When requiring high availability, configure dual servers to access shared disk storage. The shared disk supports high availability because data that the primary server writes to shared disk becomes available to the Backup server and its components during failure recovery. Because the Data Ingestion subsystem can use substantial I/O bandwidth and requires constant disk availability, Oracle Financial Services discourages the use of NFS-mounted disk storage.

The following sections describe the Data Ingestion directory structure.

Directory Structure Descriptions

Table 19 lists important subdirectories that compose the `<INSTALL_DIR>/ingestion_manager` directory structure.

Note: Please ignore all informatica variables. CTR functionality does not require informatica processing.

Table 19. Directory Structure Description

Directory Name	Description
`bin`	Contains the programs that interface with the Market data feed to capture Market data and to stream that data to the MDS (refer to *bin Subdirectory*, on page 43, for more information).
`config`	Contains files used to configure the Data Ingestion components (refer to *config Subdirectory*, on page 46, for more information).
`data/backup`	Contains backup files for the various Data Ingestion components (refer to *data/backup Subdirectory*, on page 65, for more information).
`data/errors`	Contains error files for various Data Ingestion components (refer to *data/errors Subdirectory*, on page 63, for more information).
`data/firm`	Contains Oracle Financial Services client data files that Data Ingestion components write (refer to *data/firm Subdirectory*, on page 65, for more information).
`data/market`	Contains market data files that Data Ingestion components write (refer to *data/market Subdirectory*, on page 64, for more information).
`inbox`	Contains data files that the Oracle Financial Services client provides (refer to *inbox Subdirectory*, on page 66, for more information).

Table 19. Directory Structure Description

Directory Name	Description
`informatica`	Identifies the root directory for Informatica components and directories. As part of the installation process, the system moves files for Informatica ingestion components to appropriate directories.
`jars`	Contains the Java Archive (JAR) files to run Java Data Ingestion components implemented in Java (refer to *bin Subdirectory*, on page 43, for more information).
`logs`	Contains log files that Data Ingestion components write (refer to *logs Subdirectory*, on page 67, for more information).
`scripts`	Contains all the shell scripts for running Data Ingestion components (refer to *scripts Subdirectory*, on page 43, for more information).
`tibspool`	Contains the `in` and `backup` subdirectories to organize the raw Market data files that the system creates from raw data that it extracts from the Market data feed (refer to *tibspool Subdirectory*, on page 66, for more information).

bin Subdirectory

The `bin` subdirectory within the `ingestion_manager` directory contains the programs that interface with the Market data feed to capture market and business client data and to stream that data to the Market Data server. A run script in the `scripts` subdirectory launches each program (refer to *scripts Subdirectory,* for more information).

jars Subdirectory

The `jars` subdirectory within the `ingestion_manager` directory contains Java programs that Ingestion Manager uses. A run script in the `scripts` subdirectory launches each program (refer to *scripts Subdirectory,* for more information).

scripts Subdirectory

The `scripts` subdirectory within the `ingestion_manager` directory contains the UNIX Bourne Shell scripts to run and stop runtime components. Executing a run script runs a new instance of a component. Executing a stop script terminates an active runtime component that is running in polling mode. Each script returns a termination status code.

If an application component terminates successfully, a script returns a zero return code. If the component fails to terminate successfully, the script returns a non-zero status (normally 1). Table 20 defines the run scripts for starting and stopping each component, and any special instructions.

Table 20. Run or Stop Scripts by Component

Script Names	Description or Special Instructions
`runDP.sh <data type>`	Launches an instance of the data preprocessor (`runDP.sh`). After receiving a soft-kill, the preprocessor terminates after it finishes preprocessing the data that is currently in its memory. If you configure the Preprocessor to run in batch mode, you cannot use the `stopDP.sh` script. For example: `runDP.sh` Customer To run or stop a specific Data Preprocessor, specify a valid input component that the run or stop script recognizes. If the script does not recognize the input component, it exits with an error and identifies the valid list of parameters. **Note:** A Data Preprocessor that you configure to run without polling (that is, batch mode) stops automatically when no data remains for processing. However, running a `stopDP.sh` script does not terminate batch processing.
`runDL.sh <data type>`	Launches an instance of the data loader (`runDL.sh`). You can configure the data loader to stop when it loads queued data for loading, or to poll periodically until explicitly stopped with the `stopDL.sh` script. For example: `runDL.sh` Customer To run or stop a specific data loader, specify a valid component that the run or stop script recognizes. If the script does not recognize the component, it exits with an error and identifies the valid list of parameters. **Note:** A data loader that you configure to run without polling (that is, batch mode) stops automatically when no data remains for processing. Running a `stopDP.sh` script does not terminate batch processing.
`env.sh`	Contains common configuration settings required to run Data Ingestion subsystem components. The `run*.sh` and `stop*.sh` scripts use this script.

The run scripts in Table 20 configure the executing environment for the Java component, and then execute it. All run scripts invoke the `env.sh` script to define environment variables that the components require. The run scripts also start the Java program with appropriate command line parameters, which Table 21 describes.

Note: Please ignore all informatica variables. CTR functionality does not require informatica processing.

Table 21. Environment Variable Descriptions

Parameter	Description
`classpath`	Directs the Java Runtime Environment (JRE) to the location of Java programs and supporting Java classes.
`Djava.security.policy`	Sets the location of the policy file that provides directory and network access rights to the component.
`NUM_SPLIT_FILES`	Specifies the degree of parallel processing for Informatica ingestion. The default is 10; the maximum is 10. **Note**: Please ignore this variable for CTR functionality as it does not require informatica processing.
`NUM_SPLIT_LINES`	Uses this parameter during the fuzzy name matching process. Behavior Detection splits names into multiple files. Although this is parameterized, Behavior Detection does not make this parameter transparent to the client. The best number of records is determined to be 50000.
`PROCESS_BANK_TO_BANK`	Enables ingestion to derive the BANK_TO_BANK field. Set the value to *N* if the client provides this field. **Note**: Please ignore this variable for CTR functionality as it does not require informatica processing.
`PROCESS_FOREIGN_FL`	Enables ingestion to derive the PROCESS_FOREIGN_FL field. Set the value to *N* if the client provides this field.
`server`	Instructs Java JRE to optimize for server-based processing. **Note**: Please ignore this variable for CTR functionality as it does not require informatica processing.
`Xms<NNNN>*`	Indicates the minimum number of megabytes (as `NNNN`) to reserve for Java memory allocation.
`Xmx<NNNN>*`	Indicates the maximum number of megabytes (as `NNNN`) to reserve for Java memory allocation. **Note:** Setting `Xmx` to too small a size may result in component failure.
`ACCT_TRUST_FROM_CUST`	Indicates whether the account risk should be exempt or trusted based on the exempt or trusted status of the customer's risk. The default value Y.

* Default values that are appropriate to the operating system in use (for example, Linux or Solaris) are automatically set in the `env.sh` file:

- For 64-bit operating systems, the maximum value should not be greater than 3500 MB.

- For 32-bit operating systems, the maximum value should not be greater than 1800 MB.

Minimum values vary by component; the `env.sh` file specifies these values.

config Subdirectory

The `config` subdirectory within the `data_ingest` directory contains the application configuration files, as Table 22 describes:

- DataIngestCustom.xml (refer to section *Data Ingest Custom XML Configuration File*, on page 46, for more information).
- DataIngest.properties (refer to section *Data Ingest Properties Configuration File*, on page 47, for more information).
- DataIngest.xml (refer to section *Data Ingest XML Configuration File*, on page 49, for more information).

The `DataIngest.properties` and `DataIngest.xml` files contain settings for IP addresses, port numbers, file paths, file extensions, and other runtime settings including an application's performance tuning parameters. Property files within the `config` subdirectory contain database user IDs and encrypted passwords.

The `config/datamaps` subdirectory also contains XML data maps for parsing input data and mapping processed data to fields in files and in databases. The XML data maps are preset and do not require any modifications.

Table 22. Application Configuration Files

File Name	Description
`DataIngest.properties`	Property file that contains settings that are configured at installation. These settings are of the most interest to an Oracle Financial Services client regarding modification (refer to Table 23).
`DataIngest.xml`	XML configuration file that contains settings that normally remain as is (refer to Table 24).
`DataIngestCustom.xml`	XML configuration file that contains overridden settings from `DataIngest.xml`.

The following sections describe each of these configuration files.

Data Ingest Custom XML Configuration File

The client can modify the `DataIngest.xml` file to override default settings that the system provides. However, this file is subject to change in future Oracle Financial Services releases. Therefore, upon installation of a newer Oracle Financial Services version the client must reapply any modifications in the current `DataIngest.xml` file to the newer `DataIngest.xml` file.

To simplify this process, the `DataIngestCustom.xml` file is available for use. This file holds all site-specific changes to the `DataIngest.xml` file. The client can override any settings in `DataIngest.xml` by placing the modifications in `DataIngestCustom.xml`. After installing a newer Oracle Financial Services version, the client can copy the older `DataIngestCustom.xml` file to `DataIngestCustom.xml` in the new installation.

Data Ingest Properties Configuration File

Table 23 describes the parameters for the `DataIngest.properties` configuration file.

Table 23. DataIngest.properties File Configuration Parameters

Property Name	Description	Example
`DB.Connection.URL`	Database URL for JDBC connections made by Java ingestion components. The content and format of this value is specific to the database vendor and the vendor database driver.	jdbc:oracle:oci8:@D1O9L2
`DB.Connection.Server`	Database server on which the database software is executing. This parameter is required in some circumstances where the database URL is not sufficient for the database driver software to connect to the database.	db1.clientname.com
`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.User`	Database user name that Java ingestion components uses when connecting to the database. The database user must have been assigned the appropriate privileges that Data Ingestion requires for interacting with the database.	INGEST_USER
`DB.Connection.Password`	Password that Java Ingestion components use when connecting with the database. This is set by the Password Manager Utility.	
`DB.Type`	The type of database being used.	Oracle
`MANTAS.DBSchema`	Schema name for the Mantas database schema. Typically, an Oracle Financial Services client uses the default name of “MANTAS.” Data Ingestion accesses the MANTAS schema when allocating sequence IDs to ingested records.	MANTAS
`MARKET.DBSchema`	Schema name for the MARKET database schema. Typically, an Oracle Financial Services client uses the default name of “MARKET.” Data Ingestion stores market data related records in the MARKET schema.	MARKET
`BUSINESS.DBSchema`	Schema name for the BUSINESS database schema. Typically, an Oracle Financial Services client uses the default name of “BUSINESS.” Data Ingestion stores market data related records in the BUSINESS schema.	BUSINESS
`XDP.AccountProfitAndLoss.TargetDir`	Name of the source files directory for the Data Ingestion informatica installation. Java ingestion places some files that Informatica mappings require into this directory.	/software/informatica/PC 811/SrcFiles

Table 23. DataIngest.properties File Configuration Parameters (Continued)

Property Name	Description	Example
`MDS.Adapter.RvSession.Service`	Service name for the TIBCO live market feed. Only Oracle Financial Services clients who opt to use the queue adapter to process live market data use this parameter.	7602
`MDS.Adapter.RvSession.Network`	Network name for the TIBCO live market feed. Only Oracle Financial Services clients who opt to use the queue adapter to process live market data use this parameter.	eri0
`MDS.Adapter.RvSession.Daemon`	The *daemon* parameter that processing the TIBCO live market feed requires. Only Oracle Financial Services clients who opt to use the queue adapter to process live market data use this parameter.	tcp:7602

Data Ingest XML Configuration File

Table 24 describes the parameters for the `DataIngest.xml` configuration file.

Caution: Default values for properties in this file are suitable for most deployments. Use caution when changing any default values.

Table 24. DataIngest.xml File Configuration Parameters

Property Name	Description	Example
ProcessingBatch: Specifies batch settings that override settings in the database. Overrides are primarily useful during testing.		
`ProcessingBatch.Name`	Sets the current batch name. Ingestion components process only input files that contain this value in the batch name portion of the file name. This property should be blank during normal operation.	
`ProcessingBatch.Date`	Sets the current processing date. Ingestion components process only input files that contain this value in the processing date portion of the file name. This property should be blank during normal operation. The date format is YYYYMMDD.	
`ProcessingBatch.Last`	Identifies the flag that indicates processing of the last batch of the day to Data Ingestion. This property should be blank during normal operation.	
Miscellaneous		
`DefaultSourceSystem.value`	Indicates the default value to use for source system when manufacturing reference data records.	`MTS`
`BufferSize.value`	Specifies the buffer size in kilobytes for I/O byte buffers that the MDS and FDT processes create to read input files. Use care when changing this parameter due to impact on performance and memory requirements.	`1024`
`DirectBufferSize.value`	Specifies the buffer size in kilobytes for Java NIO direct byte buffers that the MDS, MDT, and FDT processes create to read input files. Use care when changing this parameter due to impact on performance and memory requirements	`1024`
`DefaultCurrency.value`	Indicates the value to use as the issuing currency when manufacturing security records from order or trade execution records.	`USD`
`UseDirectBuffers.value`	Specifies whether to make use of Java NIO's direct buffer mechanism.	`TRUE`
Log: Specifies properties used to configure the common logging module.		

Table 24. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`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.UseDateLog`	Specifies whether to put default log file for a component in a date subdirectory. Otherwise, it is placed directly under the logs directory.	`TRUE`
`Log.InitDir`	Specifies the location of the properties file for configuring the common logging module (`install.cfg`).	`../config`
DB: Specifies properties related to database access.		
`DB.Connection.Driver`	Indicates the JDBC driver class name.	`oracle.jdbc.driver.OracleDriver`
`DB.Connection.InitialConnections`	Specifies the number of connections initially to allocate in the connection pool.	`1`
`DB.Connection.MaximumConnections`	Indicates the maximum number of connections in the connection pool. You should correlate this parameter to the number of configured threads for the component.	`10`
`DB.Connection.Timeout`	Identifies the number of seconds to wait before timing out on a database connection attempt.	`10`
`DB.Connection.NumRetries`	Specifies the maximum number of times to attempt to connect to a database before failing.	`5`
MARKET: Specifies properties related to data loaded into the MARKET schema.		
`MARKET.ExtractDir`	Specifies the parent directory for directories where the MDS component stores intermediate market data files.	`../data/market/extract`
`MARKET.TransformDir`	Specifies the directory where the MDT component stores intermediate market data files.	`../data/market/transform`
`MARKET.LoadDir`	Identifies the parent directory for directories that store market data files prior to loading with the Java data loader component. Control files for native loaders also reside below this directory.	`../data/market/load`
BUSINESS: Specifies properties related to data loaded into the BUSINESS schema.		
`BUSINESS.ExtractDir`	Identifies the parent directory for intermediate files that preprocessors produce that are applicable to the BUSINESS schema in the database.	`../data/firm/extract`
`BUSINESS.TransformDir`	Specifies the working directory for the FDT component which transforms BUSINESS trade-related data.	`../data/firm/transform`

Table 24. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`BUSINESS.LoadDir`	Indicates the parent directory for directories that store BUSINESS schema bound data files prior to loading with the Java data loader component. Control files for native loaders also reside below this directory.	`../data/firm/load`
MANTAS: Specifies properties related to data loaded into the MANTAS schema.		
`MANTAS.ExtractDir`	Specifies the parent directory for intermediate files that preprocessors produce that are applicable to the MANTAS schema in the database.	`../data/mantas/extract`
`MANTAS.TransformDir`	Specifies the working directory for intermediate files that utilities produce that are applicable to the MANTAS schema in the database.	`../data/mantas/transform`
`MANTAS.LoadDir`	Specifies the parent directory for directories that store MANTAS schema bound data files prior to loading with the Java data loader component. Control files for native loaders also reside below this directory.	`../data/mantas/load`
Directory: Specifies properties used to define directory locations.		
`Directory.Log`	Specifies the parent directory for log file directories and log files that Java ingestion components create.	`../logs`
`Directory.Inbox`	Specifies the input directory where Java ingestion components find files that the Oracle Financial Services client submits. Processing creates subdirectories in the `/inbox` directory for each day of data, to contain a copy of the input data file.	`../inbox`
`Directory.Error`	Specifies the parent directory for error directories that contain error data files that Java ingestion components create. Each error data file contains records that were not processed due to error.	`../data/errors`
`Directory.Archive`	Specifies the parent directory for directories that contain backup copies of intermediate files that Java ingestion components create.	`../data/backup`
`Directory.Config`	Specifies the directory containing configuration files for Java ingestion server.	`../config`
`Directory.FuzzyMatcher`	Specified the directory containing files related to fuzzy matcher.	`../fuzzy_match`
`Directory.DataMap`	Specifies the directory that contains XML data map files.	`../config/datamaps`
FileExtension: Specifies properties used to define extensions for various types of files.		
`FileExtension.Log`	Specifies the file name extension for log files.	`.log`

Table 24. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`FileExtension.Checkpoint`	Specifies the file name extension for checkpoint files. Many of the Java ingestion components create checkpoint files as an aid to recovery when restarted after exiting prematurely.	`.cp`
`FileExtension.Temporary`	Specifies the file name extension for temporary files that Java ingestion components create.	`.tmp`
`FileExtension.Error`	Specifies the file name extension for error files that Java ingestion components create.	`.err`
`FileExtension.Data`	Specifies the file name extension for input data files that the Oracle Financial Services client submits. The default value of *`.dat`* is in accordance with the DIS.	`.dat`
`Separator.value`	Specifies the delimiter that separates fields in data file records.	~
Security: Specifies properties used to produce security reference data.		
Security.AdditionalColumns	Specifies additional columns of data that ingestion components need to populate when manufacturing security records.	`SCRTY_SHRT_NM, SCRTY_ISIN_ID, PROD_CTGRY_CD, PROD_TYPE_CD, PROD_SUB_TYPE_CD`
Symbol: Specifies properties used for looking up security reference data by security short name.		
`Symbol.DbTableName`	Specifies the name of the database table to use when looking up security records by security short name.	`SCRTY`
`Symbol.KeyColumn`	Specifies the column name to use when looking up security records by security short name.	`SCRTY_SHRT_NM`
`Symbol.ValueColumn`	Specifies the column to use for retrieving the Behavior Detection assigned identifier for a security.	`SCRTY_INTRL_ID`
`Symbol.Category`	Specifies the category of data for the security table. The category is a key for mapping to the database schema in which the security table resides.	`BUSINESS`
SecurityISIN: Specifies properties used for looking up security ISINs.		
`SecurityISIN.DbTableName`	Specifies the name of the table to use when looking up a security using the Behavior Detection assigned security identifier.	`SCRTY`
`SecurityISIN.KeyColumn`	Specifies the column name to use when looking up security records by Behavior Detection assigned security identifier.	`SCRTY_INTRL_ID`

Table 24. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`SecurityISIN.ValueColumn`	Specifies the column to retrieve when looking up a security using the Behavior Detection assigned security identifier.	`SCRTY_ISIN_ID`
`SecurityISIN.Category`	Specifies the category of data in which the security table resides. The category is a key for mapping to the database schema in which the security table resides.	`BUSINESS`
MDS: Specifies properties used to configure the MDS component.		
`MDS.NumberOfThreads.value`	Specifies the number of worker threads that the MDS uses when processing data.	`4`
`MDS.BufferSize.value`	Allows an override to the `BufferSize.value` property for MDS.	`1024`
`MDS.Adapter.value`	Specifies the type of market data feed being provided by the client. “Reuters” indicates processing of a live market feed using TIBCO software. “File” indicates that the Oracle Financial Services client provides market data as flat files in accordance with the DIS.	`Reuters`
MDS Properties for Use with Live Market Feed		
`MDS.QueueAdapter.InputDir`	Specifies the directory that MDS should examine when looking for live market data files that TibSpool produces.	`../tibspool/in`
`MDS.QueueAdapter.BackupDir`	Specifies the directory to which MDS moves live market data files after processing them.	`../tibspool/backup`
`MDS.QueueAdapter. InputFilePrefix`	Specifies the file name prefix for live market data files that TibSpool creates.	`tib`
`MDS.DataFeed. ExchangeQuoteTimeFields`	Specifies the name of the exchange quote time field in a live market feed.	`EXCHTIM`
`MDS.DataFeed. MarketMakerQuoteTimeFields`	Specifies the name of the market maker quote time fields in a live market feed.	
`MDS.DataFeed. RICExchangeCodes`	Specifies a set of mappings of RIC exchange codes to Behavior Detection exchange codes.	`N-XNYS,B-XBOS,C-XCIS, P-XPSE,T-XTHM,PH-XPHL, A-XASE,MW-XCHI`
`MDS.DataFeed. FeedExchangeCodes`	Specifies a set of mappings of feed exchange codes to Behavior Detection exchange codes.	`1-XASE,2-XNYS,3-XBOS, 4-XCIS,5-XPSE,6-XPHL, 7-XTHM,8-XCHI,43-XNAS`
`MDS.TimeInterval.value`	Specifies the frequency in minutes with which MDS writes output data files when processing data from a live market feed.	`10`
`MDS.CacheSize.value`	Specifies the data cache byte size that the MDS uses.	`1000000`
`MDS.RvSession.Timeout`	Specifies the communication timeout value in seconds for the MDS when retrieving market summary information from a live market feed.	`60`

Table 24. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`MDS.MarketHours.marketTimeZone`	Specifies the time zone that the live market feed uses when reporting timestamps.	`GMT`
`MDS.MarketHours.localTimeZone`	Specifies the time zone that the local system uses.	`EST`
`MDS.DailySummary.SubscriptionWait`	Specifies the number of milliseconds to wait between subscription requests to the live market feed.	`100`
`MDS.DailySummary.LastSubscriptionWait`	Specifies the number of seconds to wait for a response for all subscription requests before rejecting subscriptions that have not received a response.	`60`
`MDS.QuoteSizeMultiplier.value`	Specifies the number to multiply quote sizes coming from the live market feed (that is, the lot size).	`100`
`MDS.MarketTimeDelay.value`	Specifies the delay in seconds to apply to market data queries to account for out-of-order data that a live market feed provides.	`30`
`MDS.HaltedCodes.value`	Specifies status codes within the market data that indicate a halt in trading.	`ND, NP, IMB, EQP, HRS, IVC, TH, INF, NDR, NPR, OHL, HAI, AIS`
`MDS.FeedUpCodes.value`	Specifies status codes within the market data that indicate that trading is active.	`0`
MDT: Specifies properties used to configure the MDT component.		
`MDT.NumberOfThreads.value`	Specifies the number of worker threads that the MDT uses when processing data.	`4`
`MDT.TickCodes.Rising`	Specifies the tick code to use when the price is rising.	`0`
`MDT.TickCodes.SameRising`	Specifies the tick code to use when the price is the same but the trend is rising.	`1`
`MDT.TickCodes.Falling`	Specifies the tick code to use when the price is falling.	`2`
`MDT.TickCodes.SameFalling`	Specifies the tick code to use when the price is the same but the trend is falling.	`3`
`MDT.MarketDataSource.value`	Specifies the source of market data. Valid values are File for file based access or RMI for access using an RMI server (not recommended for performance reasons).	`File`
`MDT.BufferSize.value`	Allows an override to the `BufferSize.value` property for MDT.	
FDT: Specifies properties used to configure the FDT component.		
`FDT.NumberOfThreads.Value`	Specifies the number of worker threads that the FDT uses when processing data.	`4`
`FDT.LowerDisplayLimit.Value`	Specifies the quantity below which orders are exempt from display.	`100`

Table 24. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`FDT.UpperDisplayLimit.Value`	Specifies the quantity above which orders are exempt from display.	`10000`
`FDT.OrderPriceLimit.Value`	Specifies the dollar value above which orders are exempt from display.	`200000`
`FDT.SequenceBatchSize.OrderState`	Specifies the batch size when retrieving sequence ids for OrderState records.	`10000`
`FDT.SequenceBatchSize.OrderEvent`	Specifies the batch size when retrieving sequence IDs for OrderEvent records (during end-of-day processing).	`1000`
`FDT.SequenceBatchSize.Order`	Specifies the batch size when retrieving sequence IDs for Order records.	`10000`
`FDT.SequenceBatchSize.Trade`	Specifies the batch size when retrieving sequence IDs for Trade records.	`10000`
`FDT.SequenceBatchSize.Execution`	Specifies the batch size when retrieving sequence IDs for Execution records.	`10000`
`FDT.SequenceBatchSize.DerivedTrade`	Specifies the batch size when retrieving sequence IDs for DerivedTrade records.	`10000`
`FDT.MarketDataSource.Value`	Specifies the source of market data. Valid values are File for file based access or RMI for access using an RMI server (not recommended for performance reasons).	`File`
`FDT.CalculateDisplayability.Value`	Specifies whether to calculate displayability states.	`FALSE`
`FDT.ExplainableCancelCodes.Value`	Specifies a comma-separated list of explainable cancellation codes.	
`FDT.BufferSize.value`	Allows an override to the `BufferSize.value` property for FDT.	
`FDT.LookForFutureEventTimes.value`		
`FDT.UsePrevailingSale.value`	Specifies whether to use the prevailing reported market sales price as an execution's expected print price when no comparable market sales occur during the order's marketable periods.	`FALSE`
Data Ingestion uses the following three parameters when calculating the expected print price for executions. A reported market sale is comparable to an execution when its size is in the same tier.		
`FDT.ExecutionSizeThresholds.FirstTierMax`	Specifies the maximum size for the first tier.	`1000`
`FDT.ExecutionSizeThresholds.SecondTierMax`	Specifies the maximum size for the second tier.	`5000`
`FDT.ExecutionSizeThresholds.ThirdTierMax`	Specifies the maximum size for the third tier. Any size bigger than this value is considered part of the fourth tier.	`10000`
Data Ingestion uses the next five parameters when calculating the marketable time with reasonable size attributes for an order. Processing divides orders into small, medium, and large based on their remaining unit quantities.		
`FDT.OrderSizeMarketability.MaxSmallSize`	Specifies the maximum size for an order to be considered small.	`1000`

Table 24. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`FDT.OrderSizeMarketability.MaxMediumSize`	Specifies the maximum size for an order to be considered medium.	`5000`
`FDT.OrderSizeMarketability.SmallMinPercentAtBest`	Specifies the minimum percent of a small order's remaining unit quantity that must be available at the best price for execution to be considered reasonable. The minimum percentage value must be represented in its decimal equivalent (for example 1.0 = 100%).	`1.0`
`FDT.OrderSizeMarketability.MediumMinPercentAtBest`	Specifies the minimum percent of a medium order's remaining unit quantity that must be available at the best price for execution to be considered reasonable. The minimum percentage value must be represented in its decimal equivalent (for example 1.0 = 100%).	`1.0`
`FDT.OrderSizeMarketability.LargeMinPercentAtBest`	Specifies the minimum percent of a large order's remaining unit quantity that must be available at the best price for execution to be considered reasonable. The minimum percentage value must be represented in its decimal equivalent (for example 1.0 = 100%).	`1.0`
`FDT.TradePurposeFilter.value`	Specifies a comma-separated list of trade purpose codes. Processing does not consider trades with one of these purpose codes in firm reference price derivations.	`IFADM, OFEA, CONB, CLNT, BTBX`
`FDT.RunBatchesSeparately.value`	Specifies whether the FDT treats batches as distinct from one another.	● `TRUE`: Three defined batches originate from different geographical areas in which the data in each batch does not overlap (that is, an execution in batch A does not occur against an order in batch B). ● `FALSE`: Processing does not separate data in each batch into a distinct time interval (that is, an event in batch A occurred at 10am and an event in batch B occurred at 9am, and batch B arrived after batch A).
`FDT.RegNMSExceptionCodes`	Identifies the Order Handling Codes that should be considered as Reg NMS executions.	`ISO, BAP, BRD, BOP, SOE, SHE`

Table 24. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`FDT.TreatLostEventsAsErrors.value`	Identifies whether lost events found by the FDT (refer to section *Rejection During the Transformation Stage*, on page 71, for a discussion of lost events) should be treated as errors (TRUE) or as lost events to be read in on the next run of FDT (false).	`TRUE`
FDT.OpenOrderFileExpected.value	Identifies whether an OpenOrder file will be provided by the client during an end of day batch (TRUE) or whether it will not be provided (FALSE).	TRUE
`FDT.NonExecutionTradePurposeCodes.value`	Specifies a comma-separated list of trade purpose codes. For Trade Execution records that refer to an Order and have one of these codes, the FDT will create a Trade record rather than an Execution record.	`CLNT, BTBX`
FDT.DeriveTradeBlotter.value	Specifies whether or not the FDT will create a TradeBlotter file.	`FALSE`
FDT.EnableMIFID.value	Identifies whether MiFid related data will be provided (TRUE) or not (FALSE).	FALSE
FDT.IgnoreFutureMarketRefs.value	Identifies whether the FDT will use Reported Market Sales records that occur later in time than a given trade when calculating the market reference price for that trade (FALSE) or not (TRUE).	FALSE
FDT.MaxFutureMarketRefCompTime.value	Specifies the number of seconds from the time a trade occurs during which any reported sales records cannot have the same price and quantity as the given trade to be considered as a market reference price. -1 means that this condition will not apply, 0 means the condition applies to reported sales with the same time, 5 means the condition applies to reported sales within 5 seconds of the trade, and so on. This parameter is only used if `FDT.IgnoreFutureMarketRefs.value = FALSE`.	-1
The next four parameters are used to generate records in the `TRADE_TRXN_CORRECTION` table, which record when a correction to a field of an execution, trade, or order occurs. The fields to be checked for corrections should be specified in a comma separated list of business field names. Business field names can be found in the corresponding XML data map file in the datamaps directory.		
`FDT.DeriveCorrectionFields.Trade`	Specifies which fields of a trade are monitored for corrections.	`UnitQuantity, PriceIssuing`
FDT.DeriveCorrectionFields.Execution	Specifies which fields of an execution are monitored for corrections.	UnitQuantity, PriceIssuing
FDT.DeriveCorrectionFields.DerivedTrade	Specifies which fields of a derived trade are monitored for corrections.	YieldPercentage, YieldMethodCode
FDT.DeriveCorrectionFields.Order	Specifies which fields of an order are monitored for corrections.	LimitPriceIssuing, UnitQuantity

Table 24. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
XDP: Specifies properties used to configure the Preprocessor (XDP) component.		
`XDP.Default.ArchiveFlag`	Specifies whether to archive data files. The system copies input files to the backup directory (TRUE) or deletes input files (FALSE).	`TRUE`
`XDP.Default.ErrorLimit`	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`
`XDP.Default.TargetDir`	Specifies the directory in which to place the resulting output file. If this is blank (the default), output files reside in the corresponding load directory (a subdirectory of `market/load` or `firm/load` depending on the schema of the data being processed).	
`XDP.Default.SequenceBatchSize`	Specifies the batch size when retrieving sequence IDs.	`100000`
`XDP.Default.AdditionalOutput`	Specifies a directory to contain the output file in addition to the target directory.	
`XDP.Default.DoFileLookups`	Specifies whether to do reference data lookups for fields that arrive as part of an input file (TRUE) or not do them (FALSE).	`FALSE`
`XDP.Default.DiscardLookupFailures`	Specifies whether to discard records that fail a reference data lookup (TRUE) or just log a message (FALSE).	`FALSE`
`XDP.Default.ValidatorClass`	Specifies the Java class used to validate records of a given data type. Use of subclasses occurs when the general functionality of AbstractValidator is not enough for a given data type.	`AbstractValidator`
`XDP.Default.AdapterClass`	Specifies the Java class used to process records of a given data type. Use of subclasses occurs when the general functionality of BaseFileAdapter is not enough for a given data type.	`BaseFileAdapter`
`XDP.Default.NumberOfThreads`	Specifies the number of worker threads to be used when preprocessing a file	`2`
`XDP.Default.BufferSize`	Allows an override to the `BufferSize.value` property for the XDP.	`100`
`XDP.Default.InputFileCharset`	Specifies the character set of the DIS input files provided by the client. Currently, the only supported character sets are those that are compatible with ASCII.	`UTF8`

Table 24. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`XDP.Default.SupplementalType`	Specifies an additional file type that a given XDP will create when it processes a file of the given type.	`TrustedPairMember`
`XDP.<Data Type>.IndexField`	Specifies the field used to create an index into an input file. Only valid for data types where AdapterClass is IndexFileAdapter.	`SecurityIdentifier`
`XDP.EmployeeTradingRestriction.DescendOrgTree`	When processing EmployeeTradingRestriction records, specifies whether to descend an organization's entire tree when creating records from an organization.	`FALSE`
`XDP.Account.DeriveAccountToPeerGroup`	When processing Account records, specifies whether to derive an AccountToPeerGroup record if the AccountPeerGroupIdentifier field is populated.	
`XDP.EmployeeTradingRestriction.DescendOrgTree`	If an Employee Trading Restriction record contains an Organization Identifier, then it specifies: ● Whether to create Employee Trading Restriction records for all employees in the organization and all the related child organizations defined in the Organization Relationship file (TRUE) or ● Whether to create records only for employees in the specified organization (False).	`FALSE`
`XDP.<Data Type>.<Property>`	Overrides the given property for the given Preprocessor instance.	
XDL: Specifies properties used to configure the Data Loader (XDL) component.		
`XDL.Default.Refresh`	Is valid for data types that have a load operation of *Overwrite* as defined in the DIS. This parameter specifies replacement of the entire table (TRUE) or provision of deltas (FALSE).	`TRUE`
`XDL.Default.DataFileExts`	Specifies the possible file extensions for an input file.	`.XDP, .FDT, .MDT`
`XDL.Default.CommitSize`	Specifies the number of records to update or insert before committing (not used when Direct=TRUE).	`500`
`XDL.Default.ErrorLimit`	Specifies the number of rejected records to allow before exiting with an error. If left blank (the default), processing sets no limit.	

Table 24. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`XDL.Default.DbErrorCodes`	Specifies a comma-separated list of database vendor-specific error codes that indicate data level errors (for example, data type and referential integrity). This results in rejection of records with a warning instead of a fatal failure.	`1, 140, 014, 011, 407, 140, 000,  000, 000, 000, 000, 000, 000`
The following properties apply only to the Oracle adapter.		
`XDL.Default.MaxBindSize`	Specifies the maximum number of bytes (integer) to use in the bind array for loading data into the database.	`4194304`
`XDL.Default.Direct`	Specifies whether to use direct path loading (TRUE) or conventional path loading (FALSE).	`FALSE`
`XDL.Default.Parallel`	Specifies whether a direct path load will be done in parallel (TRUE). This will be the case when multiple loaders for the same data type are run in parallel, such as with multiple ingestion instances.	`FALSE`
`XDL.Default.Unrecoverable`	Specifies whether a direct path load does not use redo logs (TRUE) or uses redo logs (FALSE).	`FALSE`
`XDL.Default.Partitioned`	Specifies whether a direct path load uses the current date partition (TRUE) or any partition (FALSE).	`FALSE`
`XDL.Default.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 Data Loader.	`FALSE`
`XDL.Default.SkipIndexErrorCode`	Specifies a database vendor specific error code that occurs in the log file when skipping indexes.	`26025`
`XDL.Default.IndexParallelLevel`	Specifies the parallel level of an index rebuild (that is, number of concurrent threads for rebuilding an index).	`4`
`XDL.Default.DoAnalyze`	Specifies whether to run a stored procedure to analyze a database table after loading data into it.	`FALSE`
`XDL.Default.DoImportStatistics`	Specifies whether to run a stored procedure to import statistics for a database table after loading data into it.	`FALSE`
`XDL.Default. ImportStatisticsType`	Specifies the type of statistic import to perform if DoImportStatistics has a value of True.	`DLY_POST_LOAD`

Table 24. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`XDL.Default.ImportStatisticsLogDir`	Saves the directory to which the stored procedure writes the log file if DoImportStatistics has a value of True. This log directory must reside on the server that hosts the database.	`/tmp`
`XDL.Default.TableDoesNotExistErrorCode`	Specifies the database error code that indicates a database table does not exist.	`942`
`XDL.Default.UseUpdateLoader`	Specifies whether JDBC updates should be used instead of a delete/insert when updating a database record. This is only valid for data types that have a load operation of Update.	`FALSE`
`XDL.<Data Type>.<Property>`	Overrides the specified property for a given Data Loader instance.	
IMC: Specifies properties for configuring the Directory Cleanup (IMC) component.		
`Directory[1..N].Name`	Identifies the directory to clean up. The system removes date subdirectories (in *YYYYMMDD* format) from this directory.	`../data/backup`
`Directory[1..N].DaysToKeep`	Specifies the number of days to keep for this directory. The system does not delete date subdirectories with the latest dates.	`3`
DBUtility: Specifies properties used to configure various utility processes. Valid utility names are SecurityMarketDaily, SecurityFirmDaily, PortfolioManagerPosition, AccountChangeLogSummary, CustomerChangeLogSummary, AccountToCustomerChangeLogSummary, CorrespondentBankToPeerGroup.		
`<UtilityName>.NumberofThreads`	Specifies the number of worker threads that the give component uses when processing data.	`4`
`<UtilityName>.SequenceBatchsize`	Specifies the batch size when retrieving sequence IDs for records generated by given component.	`10000`
Watch List Service: Specifies properties used to configure the Scan Watch List Web Service.		
`Timeout.value`	Specifies how many seconds a call to the Watch List Service made through the `scanWatchList.sh` script will wait for the service request to finish. This value should be set to the longest wait time expected based on the volume of data and system configuration. Setting it very high will not affect performance since the call will return as soon as it is complete.	`600`
`Log.UseDateLog`	Overrides the default Log.UseDateLog property.	`FALSE`
`WatchListScannerClass.value`	Identifies the Java class used to scan a watch list for a given name.	`MantasWatchListScanner`
`NameMatcherClass.value`	Identifies the Java class used to match a name against a list of names.	`FuzzyNameMatcher`

Table 24. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`FuzzyMatcher.SecondToPoll`	Identifies the number of seconds to wait between querying the WATCH_LIST table for new names that are added by the Watch List Management Utility.	
`FuzzyMatcher.MaximumAddedNames.value`	Identifies the maximum number of names that can be added to the Watch List Service after it is initialized. If additional names need to be added, the service needs to be re-initialized.	
`SSAParams.System`	This corresponds to the Informatica Identity Resolution *system* to be used by Oracle Financial Services. In practical terms, this corresponds to the name of a subdirectory under the `pr` directory in the Informatica Identity Resolution installation. For the purposes of Behavior Detection, any name can be used, but default is the standard.	`Standard`
`SSAParams.Population`	Specifies the Population rule set to be used. This generally corresponds to a Country/Language rule set (for example, Australia, Brazil, UK, and USA). The name of the population corresponds to the name of the file provided by Informatica Identity Resolution that contains the rules for matching names in the given language.	`aml.ysp`
`SSAParams.PersonalKeyLevel`	Specifies the Key Level to be used when generating Informatica Identity Resolution Keys for personal names. Standard Keys overcome more variation than Limited Keys while using less disk space than Extended Keys.	`Standard, Extended, or Limited`
`SSAParams.BusinessKeyLevel`	Specifies the Key Level to be used when generating Informatica Identity Resolution Keys for business names. Standard Keys overcome more variation than Limited Keys while using less disk space than Extended Keys.	`Standard, Extended, or Limited`
`SSAParams.PersonalSearchLevel`	Used in defining the type of Search Strategy to use when searching for personal names. The four possible values allow adjustment to the *thoroughness* of the search. The wider the search, the more candidates are typically returned, which may increase the reliability of the search; however, it uses more resources and take longer.	`Narrow, Typical, Exhaustive, or Extreme`

Table 24. DataIngest.xml File Configuration Parameters (Continued)

Property Name	Description	Example
`SSAParams.BusinessSearchLevel`	Used in defining the type of Search Strategy to use when searching for business names. The four possible values allow adjustment to the *thoroughness* of the search. The wider the search, the more candidates are typically returned, which may increase the reliability of the search; however, it uses more resources and take longer.	`Narrow, Typical, Exhaustive, or Extreme`
`SSAParams.PersonalMatchLevel`	Used in defining the level of Matching to be performed when searching for personal names. The three possible values allow adjustment to the *tightness* of the match.	`Conservative, Typical, or Loose`
`SSAParams.BusinessMatchLevel`	Used in defining the level of Matching to be performed when searching for business names. The three possible values allow adjustment to the *tightness* of the match.	`Conservative, Typical, or Loose`
`SSAParams.NumberOfQueryObjects`	Specified the number of Informatica Identity Resolution sessions open to service requests. Each session requires it's own memory area and is used to service single name matching request.	`10`
`SSAParams.NumberOfInitThreads`	Specifies the number of threads that are used to initialize the SSA_PERSONAL_NAME and SSA_BUSINESS_NAME tables that contain Informatica Identity Resolution keys corresponding to names in the WATCH_LIST table.	`10`

data Subdirectory

The `data` subdirectory within the `data_ingest` directory contains additional subdirectories for organizing Market data files and Oracle Financial Services client data files. The system creates these files during the preprocessing, transformation and data-loading stages of the ingestion process. The Market data and Oracle Financial Services client data files appear in subdirectories that are indicative of the processing stages (or workflow steps) that the Data Ingestion subsystem components perform. The following sections describe the contents of each subdirectory and the components that read or write to each subdirectory.

data/errors Subdirectory

The `errors` subdirectory within the `data` subdirectory stores error files that Data Ingestion subsystem components create or move upon detection of errors during file processing. The system places error files in subdirectories within the `errors` subdirectory. These error file subdirectories are name-based on the processing date for the files that they contain. The date has the format `YYYYMMDD`, where `YYYY` is the four-digit year, MM is the two-digit month, and DD is the two-digit day. The files in the `errors` subdirectory have the same name as the file in which the error was detected. However, the component that identified the errors appends its extension to the end of the file.

Table 25 identifies the error file signatures that each component can output to the `errors` subdirectory.

Table 25. Error File Signatures Output by Component

Component	Error File
Preprocessor	`<data type>_*.XDP.err`
Data Loader	`<data type>_*.XDL.err`
FDT	`Order_*.FDT.err` `TradeExecution_*.FDT.err`

MDS	`InsideQuote_*.MDS.err`
	`MarketCenterQuote_*.MDS.err`
	`ReportedMarketSale_*.MDS.err`

The IMC utility, `runIMC.sh`, cleans up the `errors` subdirectory. The IMC's configuration file defines the number of days that error files age before their removal.

data/market Subdirectory

The `market` subdirectory within the data subdirectory contains the `extract`, `transform`, and `load` subdirectories that correspond directly to the workflow steps that market data moves through during Data Ingestion. The following subsections describe each subdirectory in more detail.

extract Subdirectory

The `extract` subdirectory within the `market` subdirectory contains additional subdirectories that organize preprocessed Market data. It organizes data by date (that is, `YYYYMMDD`, where `YYYY` is the four-digit year, MM is the two-digit month, and DD is the two-digit day). The MDS extracts and preprocesses market data that contains a symbol's InsideQuote, MarketCenterQuote, and ReportedMarketSale information.

The IMC component, `runIMC.sh`, determines the length of time that a Market data file remains in the subdirectory before its removal. The IMC's configuration file defines the number of days that market data files persist before removal.

transform Subdirectory

The `transform` subdirectory within the `market` subdirectory contains the MDT's checkpoint data and working files that it creates during transformation. The MDT receives preprocessed data that MDS creates, and transforms that data to create derived attributes. Processing writes the transformed data to files, and moves the files to the `load` subdirectory upon completion.

The MDT also maintains checkpoint files that allow it to recover after a failure and without the loss of data integrity—the MDT removes the files after it transforms its data successfully. Table 26 identifies the files that the MDT writes to subdirectories within the `load` subdirectory.

Table 26. Files Output by Market Data Transformer

Component	Output Data Files

MDT	`InsideQuote_*.MDT`
	`MarketCenterQuote_*.MDT`
	`ReportedMarketSale_*.MDT`

load Subdirectory

The `load` subdirectory within the `market` subdirectory contains additional subdirectories that contain preprocessed and transformed Market data ready for loading into the database. Each loader component monitors its assigned subdirectory (that is, data queue), looking for data to load into the database. A subdirectory exists for each kind of Market data that a loader moves into the database. After loading data files into the database, each loader moves the processed files to the `backup` subdirectory.

Table 27 identifies the files that each data loader reads and loads into the database.

Table 27. Files that Market Data Loaders Read and Process

Component	Input Data Files
MDT	`InsideQuote*.MDT`
MDT	`MarketCenterQuote*.MDT`
MDT	`MarketState*.MDT`
MDT	`ReportedMarketSale*.MDT`
Preprocessor	`<data type>*.XDP`

data/backup Subdirectory

The `backup` subdirectory stores files that Data Ingestion subsystem components processed and require no further processing. That is, they are considered to be in a *final* form after successful processing.

- Transformers back up files that they receive and create.
- Loaders back up files that they finished loading. Each file in the backup directory appears in a subdirectory with the date as its name. The name is in the format `YYYYMMDD`, where `YYYY` is the four-digit year, MM is the two-digit month, and DD is the two-digit day.

The IMC component, `runIMC.sh`, cleans up the `backup` subdirectory. The IMC's configuration file defines the number of days that backup files age before removal. Table 28 references the files that the system writes to the `backup` subdirectory, by component.

Table 28. Backed Up Files by Component

Component	Data Files
FDT	`*.XDP`
Data Loader	`*.XDP, *.FDT, *.MDT`

data/firm Subdirectory

The `firm` subdirectory within the `data` subdirectory contains the `extract`, `transform` and `load` subdirectories that correspond directly to the workflow steps that Firm data moves through during Data Ingestion. The following sections describe each subdirectory.

extract Subdirectory

The `extract` subdirectory within the `firm` subdirectory contains checkpoint data and working files for each preprocessor during preprocessing.

Each preprocessor also maintains checkpoint files that enable it to recover after a failure and without the loss of data integrity; an FDT removes the files after it successfully preprocesses its data. When finished, each preprocessor moves its final preprocessed files to either the `transform` subdirectory for processing by FDT, or to the `load` subdirectory for loading into the database.

The `.XDP` file type identifies files that the preprocessor creates.

transform Subdirectory

The `transform` subdirectory within the `firm` subdirectory contains the FDT's checkpoint data and working files during transformation. When finished, the FDT moves its final transformed Firm data files to the `load` subdirectories for loading into the database. The system writes the transformed data to files and then moves the files to the `load` subdirectory. The `.FDT` file type identifies the files that the FDT creates.

The FDT also maintains several checkpoint files that allow it to recover after a failure, without the loss of data integrity.

load Subdirectory

The `load` subdirectory within the `firm` subdirectory contains additional subdirectories that contain preprocessed and transformed Firm data that the system queues for loading into the database. Each loader component monitors its respective subdirectory (that is, data queue) looking for data to load into the database—a subdirectory exists for each kind of Oracle Financial Services client data that processing loads into the database. After loading data files into the database, each loader moves the processed files to the backup subdirectory.

inbox Subdirectory

The `inbox` subdirectory within the `ingestion_manager` directory is an electronic mailbox or queue in which the Oracle Financial Services client writes its data files for subsequent processing by Data Ingestion subsystem Data Preprocessor components. Each Market or Firm Data Preprocessor retrieves the file it is assigned to process from the `inbox` subdirectory and then moves the file to the appropriate extract subdirectory for preprocessing. The DIS describes the naming convention and content of each data files that an Oracle Financial Services client provides.

tibspool Subdirectory

The `tibspool` subdirectory contains files that the TibSpool component (TIBS) produces and the MDS reads. These files are in a raw Tibco binary format and contain market data messages from a live market data feed. TIBS writes files to the `in` subdirectory. The MDS reads these files from the `in` subdirectory and moves the files to the `backup` subdirectory after extracting the data from them.

tibspool/in Subdirectory

The `in` subdirectory within the `tibspool` subdirectory contains the raw data files that a TIBS instance extracts from the Market data feed. During normal processing, these data files reside in this location temporarily until deletion. Each file name has the following format:

```
tib<A-Z>_<YYYYMMDDHHMMSS>_<NNNNNN>.dat
```

where:

- `<A-Z>` is a symbol range that corresponds to the symbol range that the TIBS instance is processing.
- `<NNNNNN>` is a required sequence number to create unique file names. Table 29 identifies sample output file names using the latter format.

Table 29. Output Files from TIB Spoolers

Component	Example Output Data Files
TIBS	`tibSA-Z_20011029112338_000001.dat` `tibSA-Z_20011029112338_000002.dat`

tibspool/backup Subdirectory

The `backup` subdirectory within the `tibspool` subdirectory contains the backup of raw data files from the Market data feed.

logs Subdirectory

The `logs` subdirectory contains a log file for each component running on a host computer. Each log file in the `logs` subdirectory appears in a subdirectory with the date as its name, in the format `YYYYMMDD`, where `YYYY` is the four-digit year, MM is the two-digit month, and DD is the two-digit day. The subdirectory's date is based on the processing date for data to which the log files pertain.

The IMC utility, `runIMC.sh`, cleans up the `logs` subdirectory. The IMC utility's configuration file defines the number of days that log files age before their removal. Table 30 identifies log files for each component, based on the file name's prefix.

Table 30. Log Files Output by Component

Prefix	Component
XDP	Preprocessor
XDL	Data loader
MDS	Market Data Server
FDT	File Data Transformer
MDT	Market Data Transformer
TibW	TibSpool
TibW	TibWatch
IMC	IMC

Startup and Shutdown

This section discusses Data Ingestion subsystem startup in both normal and recovery modes. You can start and stop all components manually with their respective run and stop scripts, with the exception of some components when configured to run in *batch* mode. Given the complexity of Data Ingestion processing, Oracle Financial Services recommends that an Oracle Financial Services client use a scheduling or monitoring tool to execute the run scripts (and stop scripts, if needed) automatically, monitor each component's progress, and alert support staff if problems arise. Scheduling or monitoring tools typically invoke a job control script that executes a Data Ingestion subsystem run and stop scripts. In addition, using the distributed processing configuration startup approach varies (refer to section *Distributed Processing Configuration*, on page 69, for more information).

Backup Server Configuration

An Oracle Financial Services client can implement a backup server configuration to collect market data in parallel (that is, in duplicate) with the Primary server to help minimize the risk of losing market data for an entire day if the Primary server fails. This form of high availability drives configuration of Data Ingestion subsystem components and when to start and stop them. In a high availability configuration, the backup server transforms and loads market data when the Primary server fails or when market data that the system is collecting on the Primary server is interrupted and causes missing data gaps. Also, a backup server configuration requires that shared disk be available for checkpoint recovery.

The daily processing cycle and desired server configuration influences how and when the system starts and stops Data Ingestion subsystem components under normal conditions, and if error recovery is necessary.

Recovery

The Data Ingestion components are designed to be able to restart after a failure. Examples of failures include database, file system, network, machine, or component. After a component fails (returns a non-zero exit status), the general recovery procedure involves checking the component's log file for the cause of the error, fix that cause (restart the database, add more disk space to the file system, etc.), and restart the component (using the same command used to start it initially). Do not run components that depend on the component that failed until successful completion of the failed component.

The exception to this procedure is live market Data Ingestion using the Queue Adapter (that is, when a market feed such as Reuters is in use). If a TIBS component or the machine on which it is running fails, recovery of lost data while the component or machine is down is impossible. To address this situation, Oracle Financial Services recommends that TIBS components be run on two separate machines, connected to two separate TIBCO infrastructures. If the primary TIBS fails, ingestion can proceed with the market data that the backup TIBS produces. The Oracle Financial Services client's job scheduling software can be configured to ingest market data through the primary or backup server as needed.

Distributed Processing Configuration

An Oracle Financial Services client can implement a distributed processing configuration that can run the Data Ingestion subsystem components on two or more servers, and let each server extract, transform, and load data for non-overlapping data. This distributed computing configuration influences configuration of Data Ingestion subsystem components or when to start and stop them. The Oracle Financial Services client is responsible for splitting data into non-overlapping sets and placing this data into the inbox for each Data Ingestion instance. For trading data and market data, the client can split data by symbol ranges (for example, A through J on one server and K through Z on the other). For reference data, the client can select an arbitrary field to use in splitting the data.

Note that it is not necessary to split reference data and process it with multiple instances, in situations, where use of multiple instances processes trading and market data. If ingestion of reference data occurs across multiple instances, the client should ensure that ingestion of all reference data of a particular type occurs prior to ingesting data that is dependent on that type of data.

Data Rejection During Ingestion

The Ingestion Manager can reject records at the Preprocessing, Transformation, or Loading stages. The following sections provide an overview of the most frequent types of conditions that cause transactions to be rejected:

- **Rejection During Preprocessing Stage:** describes how rejections occur during the Preprocessing stage and offers guidance on ways to resolve rejections (refer to section *Rejection During the Preprocessing Stage*, on page 70, for more information).
- **Rejection During Transformation Stage:** describes how rejections occur during the Transformation stage and offers guidance on ways to resolve rejections (refer to section *Rejection During the Transformation Stage*, on page 71, for more information).
- **Rejection During Loading Stage:** describes how rejections occur during the Loading stage and offers guidance on ways to resolve rejections (refer to section *Rejection During the Loading Stage*, on page 73, for more information).

Rejection During the Preprocessing Stage

The first stage of ingestion is Preprocessing. At this stage, Data Ingestion examines Oracle Financial Services client reference and trading data for data quality and format to ensure the records conform to the requirements in the DIS. Common reasons for rejection of data during Preprocessing include problems with data type, missing data, referential integrity, and domain values.

During normal operation, the number of rejections at the preprocessor stage should be minimal. If the volume of rejections at this stage is high, a decision threshold can halt processing and allow manual inspection of the data. The rejections are likely the result of a problem in the data extraction process. It is possible to correct the rejections and then reingest the data.

Data Type

Every field in a record that processing submits to the Ingestion Manager must meet the data type and length requirements that the DIS specifies. Otherwise, the process rejects the entire record. For example, fields with a *Date Type* must appear in the format YYYYMMDD. Thus, the date April 30, 2005 has a format of 20050430 and, therefore, is unacceptable. In addition, a field cannot contain more characters or digits than specified. Thus, if an Order Identifier in an Order record contains more than the maximum allowed length of 40 characters, rejection of the entire record occurs.

Missing Data

The DIS defines fields that are mandatory, conditional, and optional. If a record contains a field marked mandatory, and that field has a null value, processing rejects the record. For example, all Trade Execution records must contain a Trade Execution Event Number. If a field is marked conditional, it must be provided in some cases. Thus, an Order record for a limit order must contain a Limit Price, but an Order record for a market order need not contain a Limit Price.

Referential Integrity

In some cases, you can configure Ingestion Manager to reject records that refer to a missing reference data record. For example, Ingestion Manager can reject an order that refers to a deal that does not appear in the Deal file. The default behavior is not to reject records for these reasons.

Domain Values

Some fields are restricted to contain only one of the domain values that the DIS defines. The Ingestion Manager rejects records that contain some other value. For example, Ingestion Manager rejects any Order record that contains an Account Type other than CR, CI, FP, FB, ER, IA, EE or any Special Handling Code other than that in the DIS.

Rejection During the Transformation Stage

The second stage of ingestion is Transformation. At this stage, the Ingestion Manager derives the order and trade life cycles, and other attributes, that are necessary for trade-related surveillance. The Ingestion Manager rejects order records during Transformation for the following reasons:

- New and Cancel or Replace order events if the order identifier and placement date combination already exists; order identifiers must be unique during a given day.
- New order events for child orders if the referenced parent order is itself a child order; only one level of a parent-child relationship is allowed.

The Ingestion Manager rejects trade execution records for New and Cancel or Replace trade execution events if the trade execution identifier and trade execution date combination already exists. Trade execution identifiers must be unique during a given day.

Other problems can occur that do not cause rejection of records but cause handling of the records to be different:

- Lost Events
- Out of Sequence Events

The following sections describe these issues.

Lost Events

If the system receives an order event other than a New or Cancel or Replace in a set of files before receiving the corresponding New or Cancel or Replace, it writes the order event to a lost file. The system examines events in the lost file during processing of subsequent sets of files to determine whether the system received the corresponding New or Cancel or Replace event. If so, processing of this event is normal. If an event resides in the lost file when execution of open order processing occurs (that is, execution of `runDP.sh OPEN_ORDER`), processing rejects the event. The same applies to trade execution events. In addition, if a New trade execution event references an order but the system did not receive the order, the New event also resides in the lost file subject to the same rules.

If rejection of a New or Cancel or Replace order or trade execution occurs during the preprocessor stage, all subsequent events are considered lost events. Submission of missing New or Cancel or Replace event can occur in a subsequent set of files, and processing of the lost events continue normally.

Out-of-Sequence Events

An out-of-sequence event is an order or trade execution event (other than New or Cancel or Replace) that the system processes in a set of files after processing the set of files that contains the corresponding New or Cancel or Replace event. Such an event that has a timestamp prior to the timestamp of the last event against that order or trade is considered an out-of-sequence event.

For example, File Set 1 contains the following events:

- NW order event, timestamp 09:30:00.
- MF order event, timestamp 09:45:00.

File Set 2 contains the event MF order event, timestamp 09:40:00.

This second MF event is considered out of sequence. This also applies to trade execution events against orders.

For example, File Set 1 contains the following events:

- NW order event, timestamp 09:30:00.
- MF order event, timestamp 09:45:00.

File Set 2 contains NW trade execution event (references the above order), timestamp 09:40:00.

This trade execution event is considered out of sequence. It is important to note that this also includes market data. If, in a given batch, market data up to 10:00:00 is used to derive attributes for a given order, any event in a subsequent file against that order with a timestamp prior to 10:00:00 is considered out of sequence.

An out-of-sequence event has no effect on the order or trade that it references. Processing sets the out-of-sequence flag for the event to Y(es) and the system writes the event to the database. Out-of-sequence indicators for any summaries that the event affects are set to Y(es), which indicates that potential compromise of their life cycles occurred.

For end-of-day processing, this may not be an issue. For Intra-day processing, subsequent files should contain data in an ever-increasing time sequence. That is, the first set of files should contain data from 09:00:00 to 11:00:00, the second set of files should contain data from 11:00:00 to 12:00:00, and so on. This only affects events in a single order or trade’s life cycle. For example, Batch 1 contains the following events:

- NW order event for order X, timestamp 09:30:00.
- MF order event for order X, timestamp 09:45:00.

Batch 2 contains the event NW order event for order Y, timestamp 09:40:00.

This order event is not considered out of sequence; processing continues normally.

Rejection During the Loading Stage

The last stage of ingestion is Loading. At this stage, the Ingestion Manager loads orders, executions, and trades into the database. The Ingestion Manager rejects records during Loading if configuration of the database is incorrect (for example, setup of partitions are incorrect for the data being ingested).

Data Ingestion 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 Financial Services client can run the `runIMC.sh` script to perform the Ingestion Manager 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 running match alert creator.

To Run Match Alert Creator

To run the match Alert Creator, follow the 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.

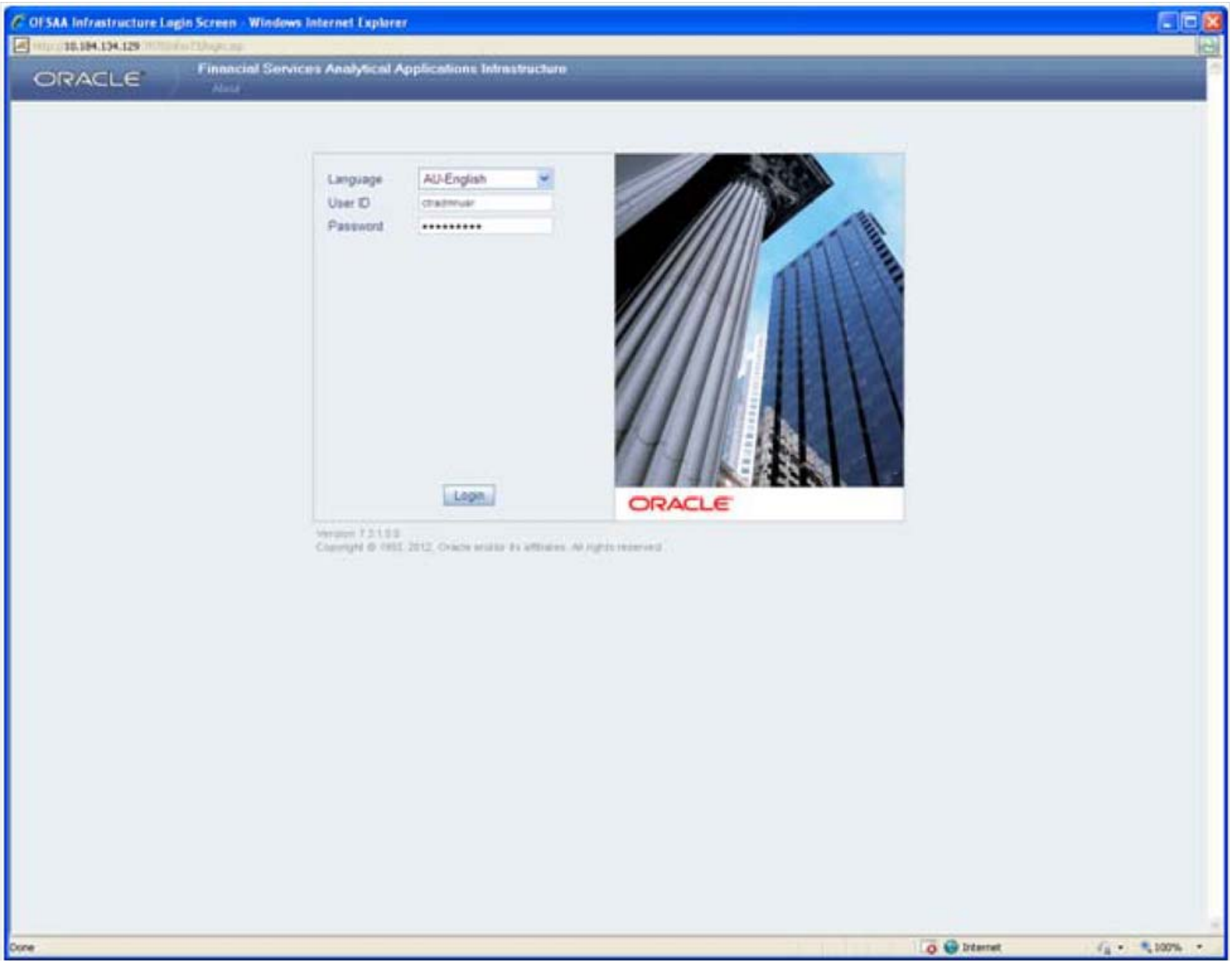
Batch Execution of CTR

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

Note: Analyse your tables before executing the CTR Batch. Refer to *Database Statistics Management,* on page 133, for more information.

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

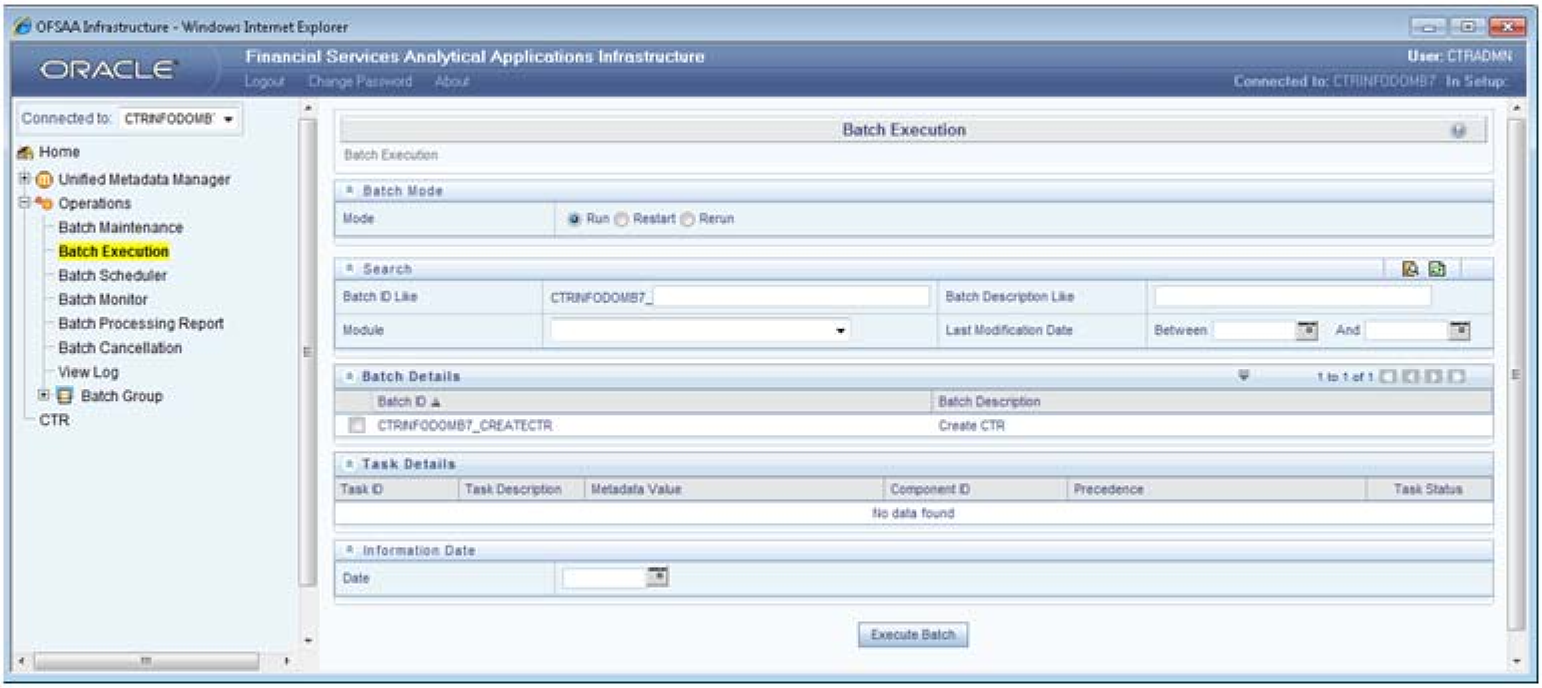
1. Login to the OFSAAI as a CTR Administrator User.


OFSAA Infrastructure Login Screen - Windows Internet Explorer
ORACLE
Financial Services Analytical Applications Infrastructure
Login
ORACLE

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

Figure 5. OFSAAI Login Screen

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



Connected to: CTRINFODOMB

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

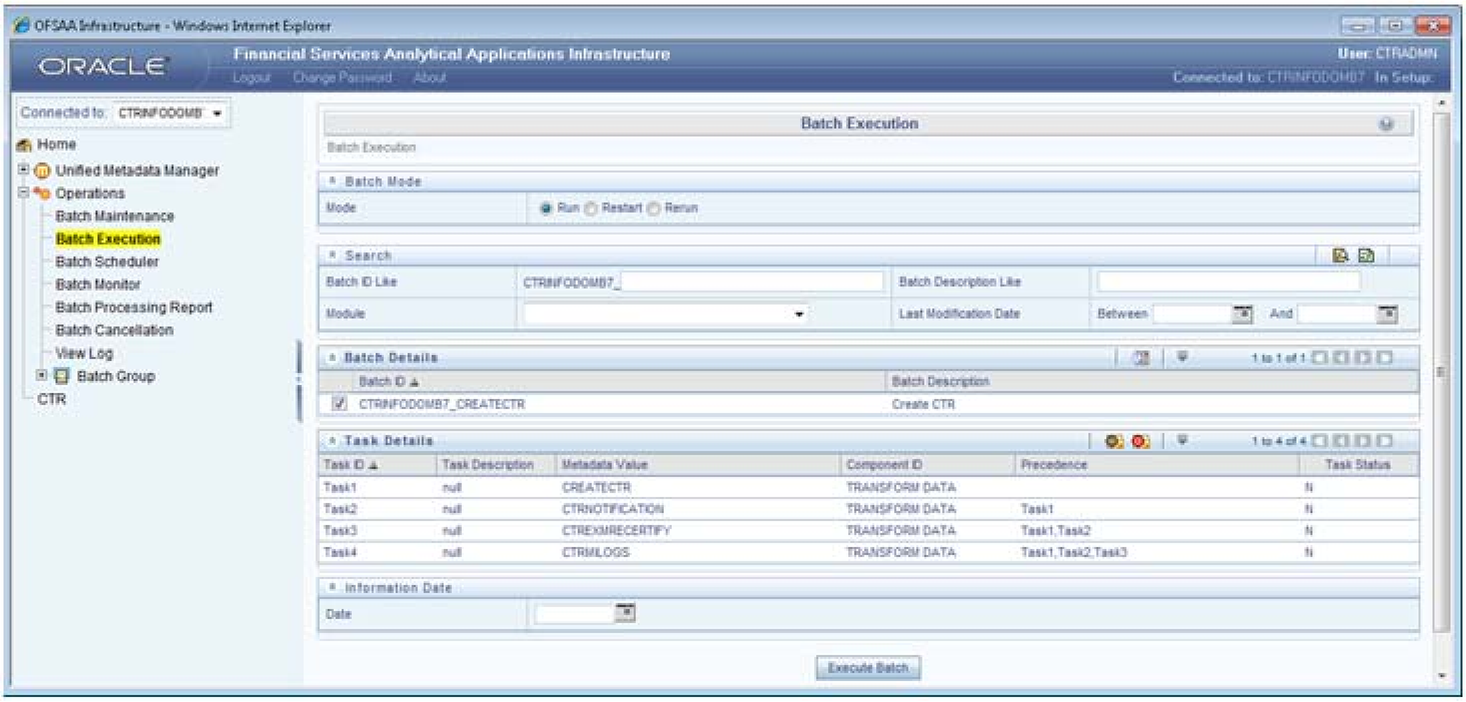
Task ID	Task Description	Metadata Value	Component ID	Precedence	Task Status
No data found					

Information Date

Date	

Figure 6. CTR Batch Execution Link

4. Select the batch (Create CTR) from batch details section.



Connected to: CTRINFODOMB

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

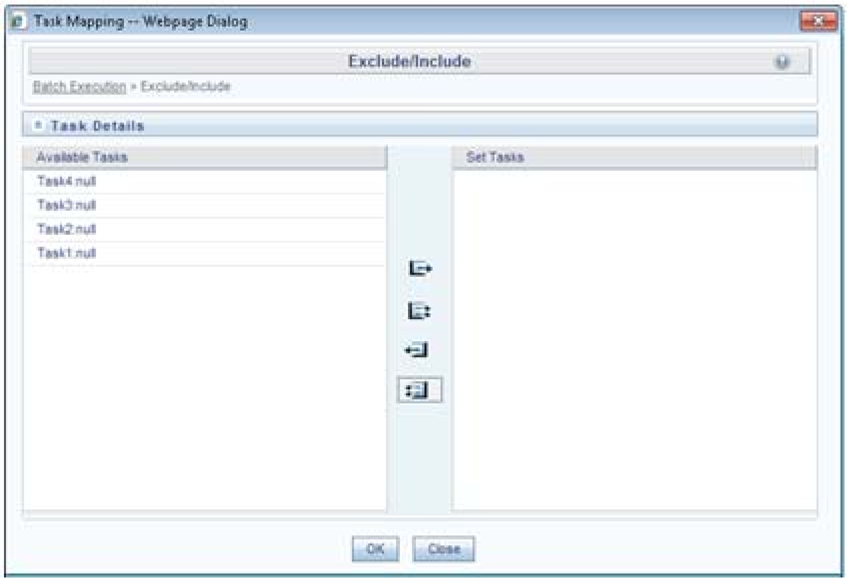
Task ID ▲	Task Description	Metadata Value	Component ID	Precedence	Task Status
Task1	null	CREATECTR	TRANSFORM DATA		N
Task2	null	CTRNOTIFICATION	TRANSFORM DATA	Task1	N
Task3	null	CTREXMRECERTIFY	TRANSFORM DATA	Task1,Task2	N
Task4	null	CTRMLOGS	TRANSFORM DATA	Task1,Task2,Task3	N

Information Date

Date	

Figure 7. CTR Batch Execution - Select Batch ID

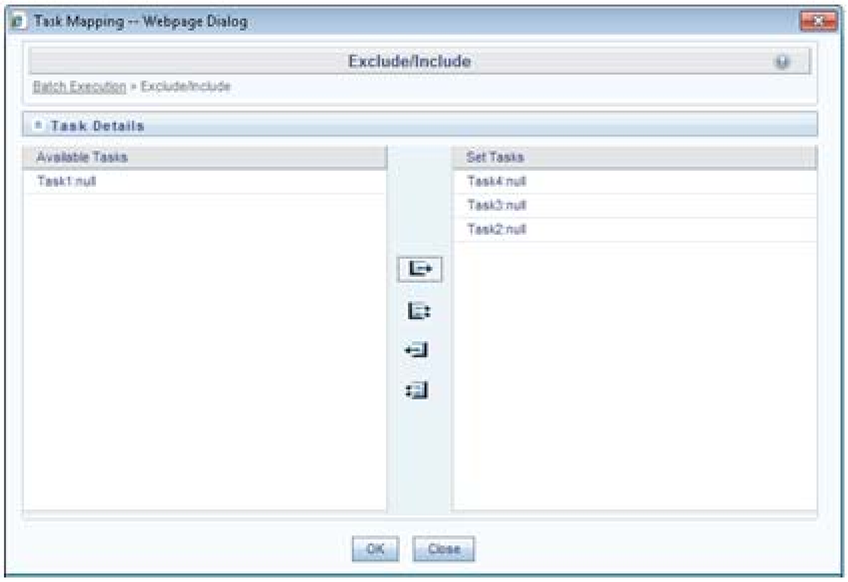
5. Click the **Exclude/Include** icon in the Task details. The Task Mapping window displays.


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 8. Test Mapping Window

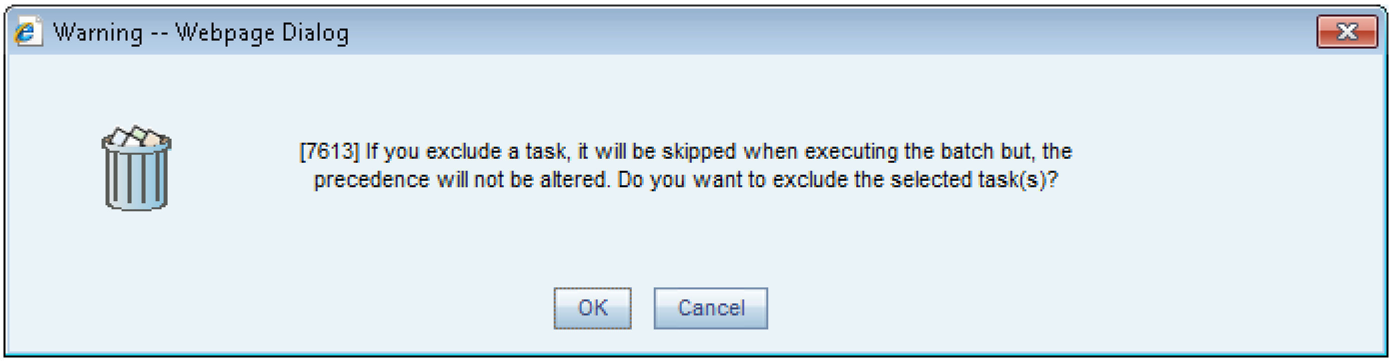
6. Keep the required tasks that you want to execute under the Available Tasks and move the rest to the Set Tasks section.


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 9. Test Mapping Window - Select Tasks

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

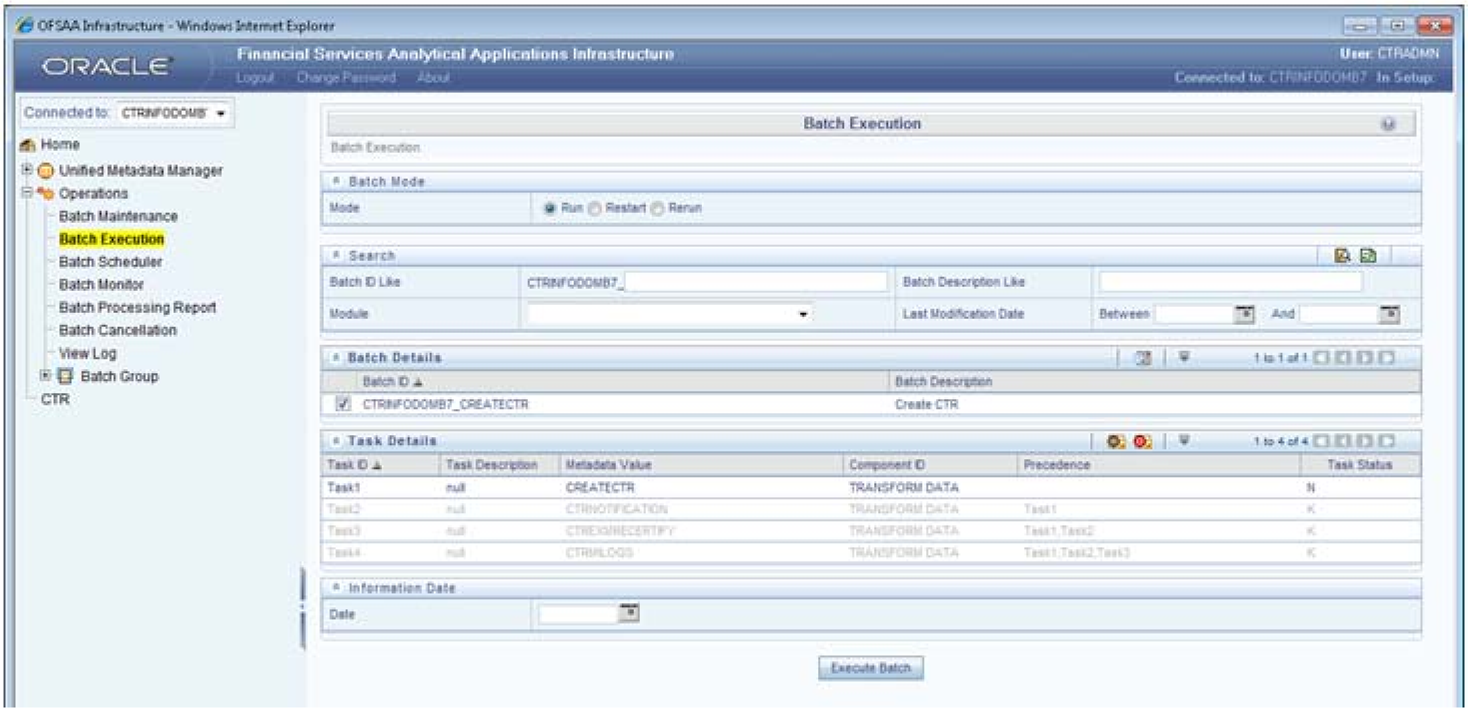


Warning -- Webpage Dialog

[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 10. Warning Message

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



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

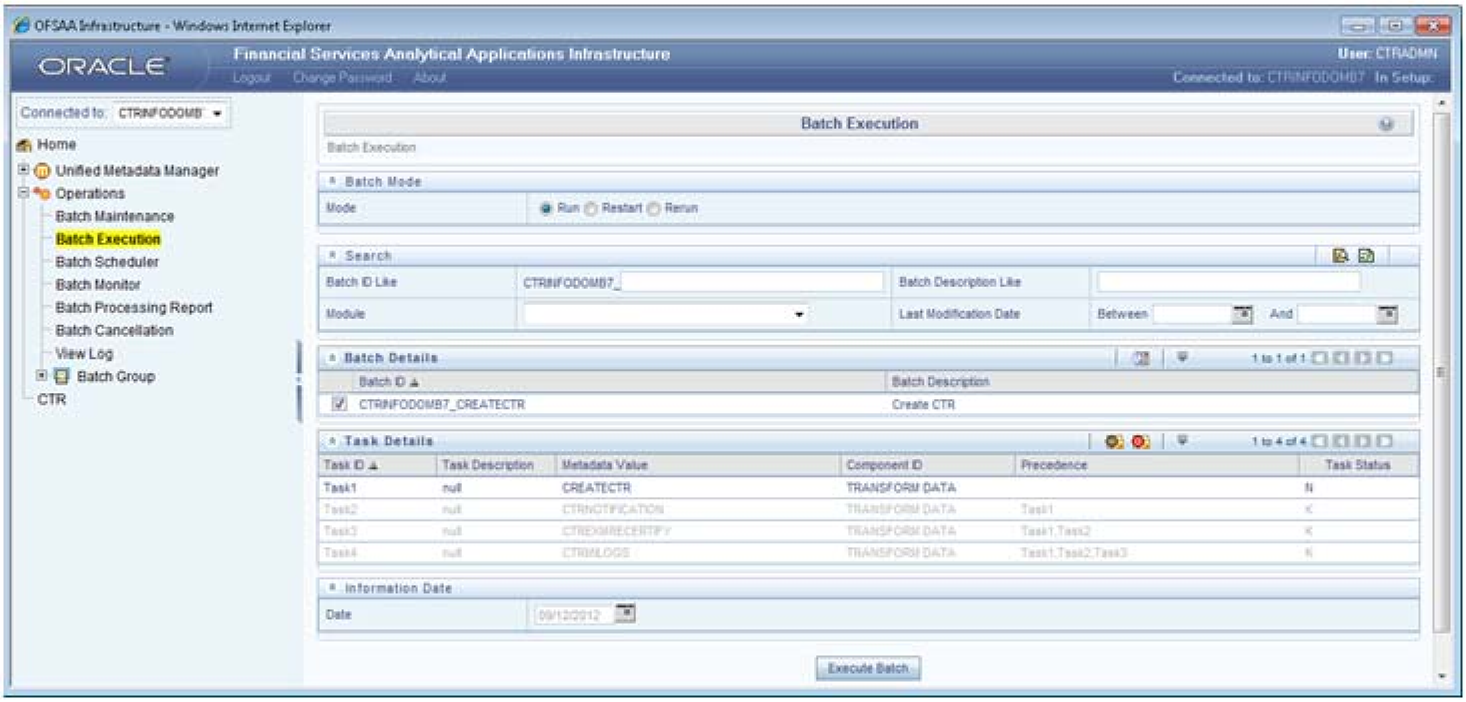
Task ID	Task Description	Metadata Value	Component ID	Precedence	Task Status
Task1	null	CREATECTR	TRANSFORM DATA		N
Task2	null	CTRNOTIFICATION	TRANSFORM DATA	Task1	K
Task3	null	CTREXMRECERTIFY	TRANSFORM DATA	Task1,Task2	K
Task4	null	CTRMLOGS	TRANSFORM DATA	Task1,Task2,Task3	K

Information Date

Date:

Figure 11. 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

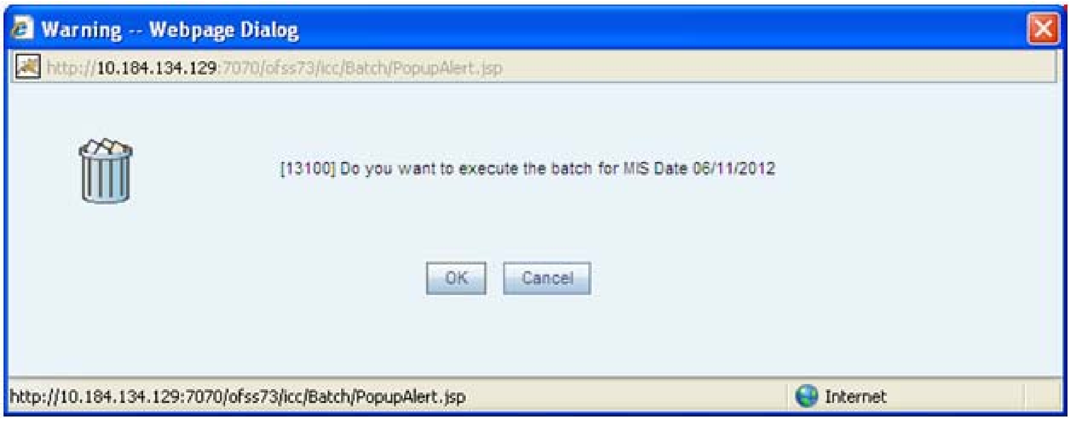
Information Date

Date: 06/12/2012

Figure 12. CTR Batch Execution - Select Information Date

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

The below window is displayed:

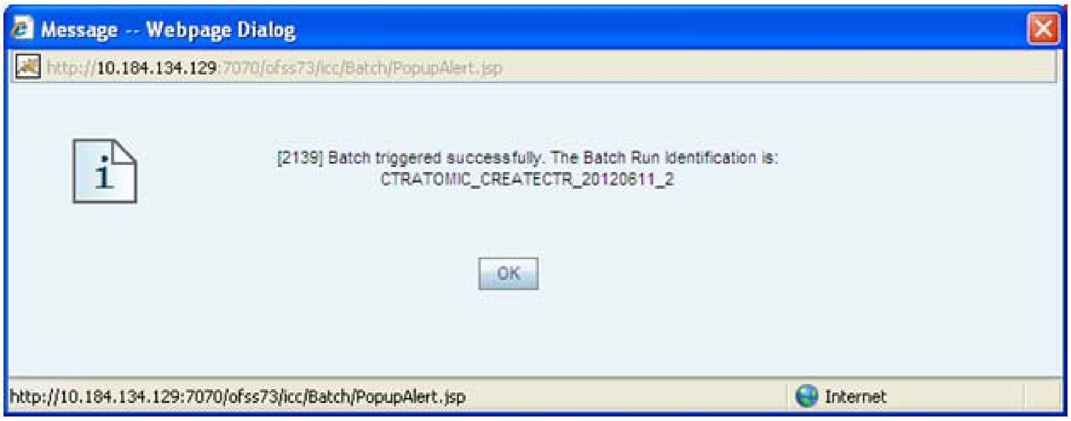

Warning -- Webpage Dialog
http://10.184.134.129:7070/ofss73/icc/Batch/PopupAlert.jsp
OK
Cancel
http://10.184.134.129:7070/ofss73/icc/Batch/PopupAlert.jsp
Internet

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

Figure 13. CTR Batch Execution Warning Window

11. Click **OK**.

A pop 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
Internet

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

Figure 14. CTR Batch Execution Confirmation Message Window

12. Click **OK**.

About CTR Web-Service Invocation

In order to invoke the CTR web service following are the 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: CTRPassCTR (default), It can be changed using utility.

Changed password must be encrypted through AES encryption algorithm with default Reveleus Key (Platform encryption key)

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 (Refer to *Alert Purge Utility*, on page 101 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 (Refer to *Batch Control Utility*, on page 109 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 (Refer to *Calendar Manager Utility*, on page 117 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 (Refer to *Data Retention Manager*, on page 122 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.

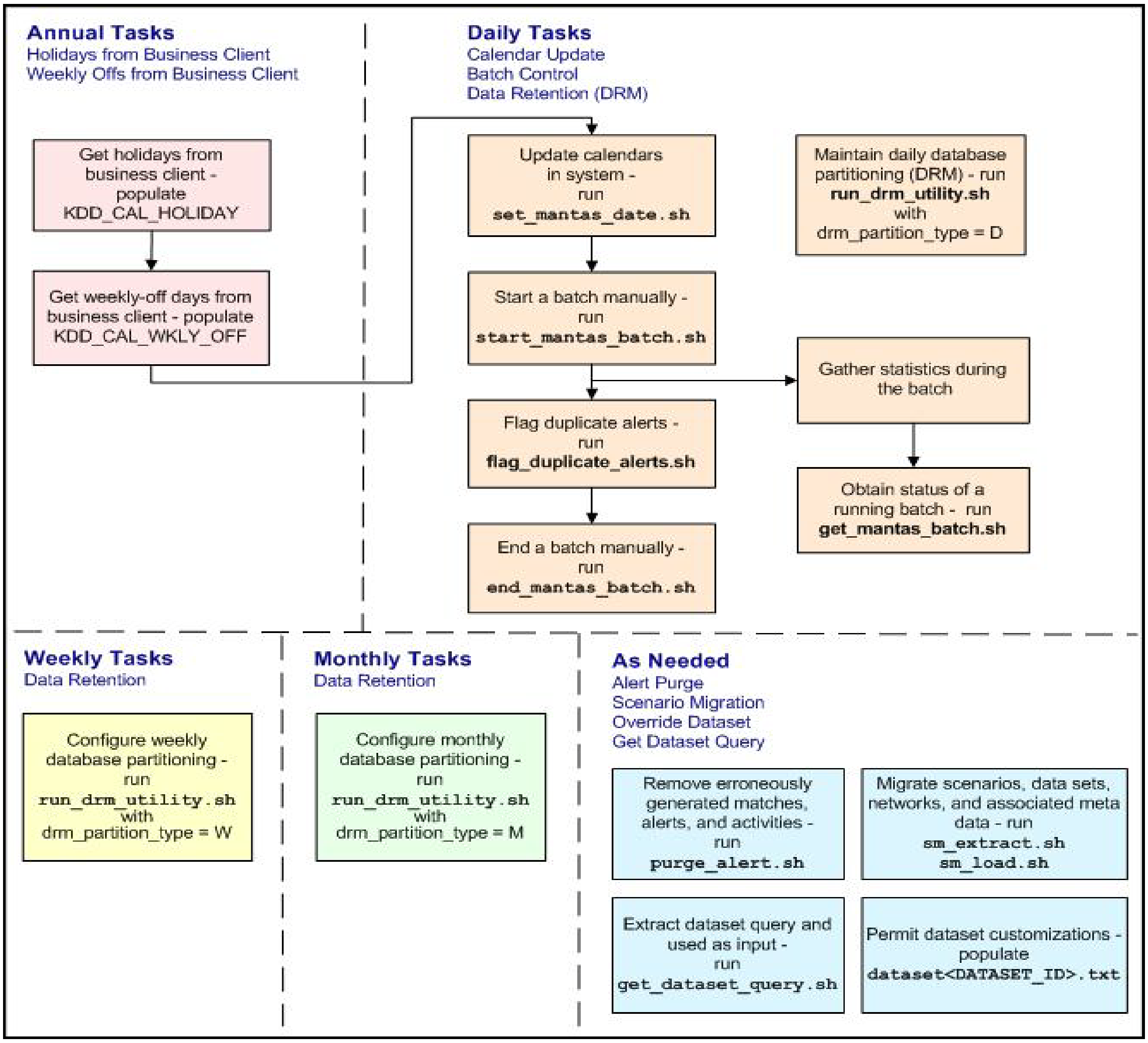

Annual Tasks
Holidays from Business Client
Weekly Offs from Business Client
Get holidays from business client - populate KDD_CAL_HOLIDAY
Get weekly-off days from business client - populate KDD_CAL_WKLY_OFF
Daily Tasks
Calendar Update
Batch Control
Data Retention (DRM)
Update calendars in system - run 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 15. 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 16 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}
```

(Continued on next page)

(Continued from previous page)

```
extract.schema.business=${schema.business.owner}
extract.schema.market=${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
```

Continued on next page)

(Continued from previous page)

```
################ 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)
#

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
```

(Continued on next page)

(Continued from previous page)

```
# 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=

# 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;
```

(Continued on next page)

(Continued from previous page)

```
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;
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=,
```

(Continued on next page)

(Continued from previous page)

```
######### Execute Query Tool CONFIGURATION #########################
#

# Username and password for connecting to the database

eqt.database.username=${ingest.user}
eqt.database.password=${ingest.password}
############# Database Builder Utility Configuration ###############
#
# File containing tokens and their value
db_tools.tokenfile=/users/mantast/Solaris10_mantas58_b09_Ti5O10S10_Iron_13080_WAS/database
/db_tools/mantas_cfg/db_variables.cfg

Oracle.DuplicateRow=1
Oracle.ObjectExists=955,2260,2275,1430,1442,1451,957,1408,2261
Oracle.ObjectDoesNotExist=942,1418,1434,2441,904,4043,1927,2443

############# Correlation Migration Utility Configuration ###############
#
corrRuleMig.CorrRuleFileNm=
corrRuleMig.loadHistory=Y
aps.service.url=http://localhost:8070/mantas/services/AlertProcessingService

############# Config Migration Utility Configuration ###############
config.filenm.prefix=Config

#################### LOG CONFIGURATION ################################
#
# Trace SQL exception.  Set to "true" for SQL tracing,
# "verbose" to trace low-level JDBC calls
#
com.sra.kdd.tools.database.debug=true

# 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=false
log.trace=false
log.time.zone=US/Eastern

# Specify whether logging for a particular level should be performed
# synchronously or asynchronously.
```

(Continued on next page)

(Continued from previous page)

```
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 filename of the message library.
log.message.library=/users/mantast/Solaris10_mantas58_b09_Ti5O10S10_Iron_13080_WAS/databas
e/db_tools/mantas_cfg/etc/mantas_database_message_lib_en.dat

# Specify the full path to the categories.cfg file
log.categories.file.path=/users/mantast/Solaris10_mantas58_b09_Ti5O10S10_Iron_13080_WAS/da
tabase/db_tools/mantas_cfg/

# 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 mantas category unless
# otherwise specified above.
# Note that if this property is not specified, logging will go to the console.
log.default.location=/users/mantast/Solaris10_mantas58_b09_Ti5O10S10_Iron_13080_WAS/databa
se/db_tools/logs/Utilities.log

# 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=/users/mantast/Solaris10_mantas58_b09_Ti5O10S10_Iron_13080_WAS/data
base/db_tools/logs/mantaslog.log
```

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 16. 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 17. Sample Logging Information in the categories.cfg File

Configuring Console Output

Figure 18 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 18. 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 18, 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

Oracle Financial Services requires that you perform certain calendar management tasks at least annually: loading holidays and weekly off-days from an Oracle Financial Services client. This ensures that Oracle Financial Services 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 19 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 19. Sample KDD_CAL_HOLIDAY Table Loading Script

Table 31 provides the contents of the `KDD_CAL_HOLIDAY` table.

Table 31. 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 Financial Services client, load this information for the upcoming calendar year into the `KDD_CAL_WKLY_OFF` table.

Figure 20 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 20. 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 32 provides the contents of the `KDD_CAL_WKLY_OFF` table.

Table 32. 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 33 provides the directory structure for the Alert Purge Utility.

Table 33. 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, Refer to *Common Resources for Administrative Utilities,* on page 84, and Appendix A, *Logging*, on page 137 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.lo`g 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 (Refer Figure 21). 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 21. 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 34 describes required and optional parameters for this utility.

Table 34. 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 table`). 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 34. 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 `'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.
`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 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 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:

- An 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 35 defines the matches that relate to these alerts and actions:

Table 35. 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 35 (Refer to section *Sample Purge Alerts Utility Run: Situation One*, on page 107 for more information).
- Sample Purge Alerts Utility Run: Situation Two shows how to purge all matches in the first job in Table 35 regardless of their alert affiliation (Refer to section *Sample Purge Alerts Utility Run: Situation Two*, on page 107 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 (Refer to section *Sample Purge Alerts Utility Run: Situation Three*, on page 107 for more information).

Sample Purge Alerts Utility Run: Situation One

To purge only those alerts that contain the first job in Table 35, 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 35, 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` (Refer to *install.cfg file,* on page 85, 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 36 provides the directory structure for the Batch Control Utility, in `<INSTALL_DIR>/database/db_tools/`:

Table 36. 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, Refer to *Common Resources for Administrative Utilities,* on page 84, and Appendix A, *Logging*, on page 137 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 (Refer to Figure 22 on page 111). 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 22. 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 (Refer to Table 44 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 37 provides the contents of the `KDD_PRCSNG_BATCH` table.

Table 37. KDD_PRCSNG_BATCH Table Contents

Column Name	Description
`PRCSNG_BATCH_NM`	Name of the batch (for example, DLY).
`PRCSNG_BATCH_DSPLY_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 38.

Table 38. `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 39 describes.

Table 39. `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. Refer to *Calendar Manager Utility,* on page 117, 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.

Refer to *Data Retention Manager,* on page 122, 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 40 describes.

Table 40. `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`. Refer to *Configuring Console Output,* on page 97, 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 37).
2. The utility determines whether a batch is running by querying the `KDD_PRCSNG_BATCH_CONTROL` table (Table 41).

Table 41. `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 42 describes the `KDD_PRCSNG_BATCH_HIST` table.

Table 42. `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 23. 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 23. 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 (Refer to Table 37 on page 111).

To End a Batch Manually

To stop a batch process manually, 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. 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 (Refer to Table 37 on page 111).
2. Records information about the batch in the `KDD_PRCSNG_BATCH_ HIST` table (Refer to Table 42 on page 114). This table contains a history of all batches that appear by start date and end date. Figure 23 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 24. 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.

To Obtain a Batch Name

To obtain a batch name, 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:

   ```
   get_mantas_batch.sh
   ```

The name of the currently running batch is written to standard output (Refer to *Configuring Console Output,* on page 97, 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 (Refer to Table 37 on page 111).
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 manage Oracle Financial Services 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 (Refer to *install.cfg file,* on page 85, for more information).

This section contains the following topics:

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

Directory Structure

Table 43 provides the directory structure for the Calendar Manager Utility, in `<INSTALL_DIR>/database/db_tools/`..

Table 43. 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 Oracle Financial Services 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, Refer to *Common Resources for Administrative Utilities,* on page 84, and Appendix A, *Logging*, on page 137, for more information.

Calendar Information

The Calendar Manager Utility obtains all holidays and weekly off-days for loading into the Oracle Financial Services calendars by retrieving information from the `KDD_CAL_HOLIDAY` and `KDD_CAL_WKLY_OFF` tables (Refer to Table 31 and Table 32). These tables contain calendar information that an Oracle Financial Services 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 (Refer to Figure 25). 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 25. 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. Refer to *Configuring Console Output,* on page 97, for more information.

Updating the `KDD_CAL` Table

As previously discussed, the Calendar Manager Utility retrieves information that it needs for updating Oracle Financial Services 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 44 provides the contents of the `KDD_CAL` table.

Table 44. `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 44. `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 (Refer to 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 45 provides the directory structure for the Data Retention Manager.

Table 45. 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, Refer to *Common Resources for Administrative Utilities,* on page 84, and Appendix A, *Logging*, on page 137 for more information.

Processing Flow

Figure 26 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.

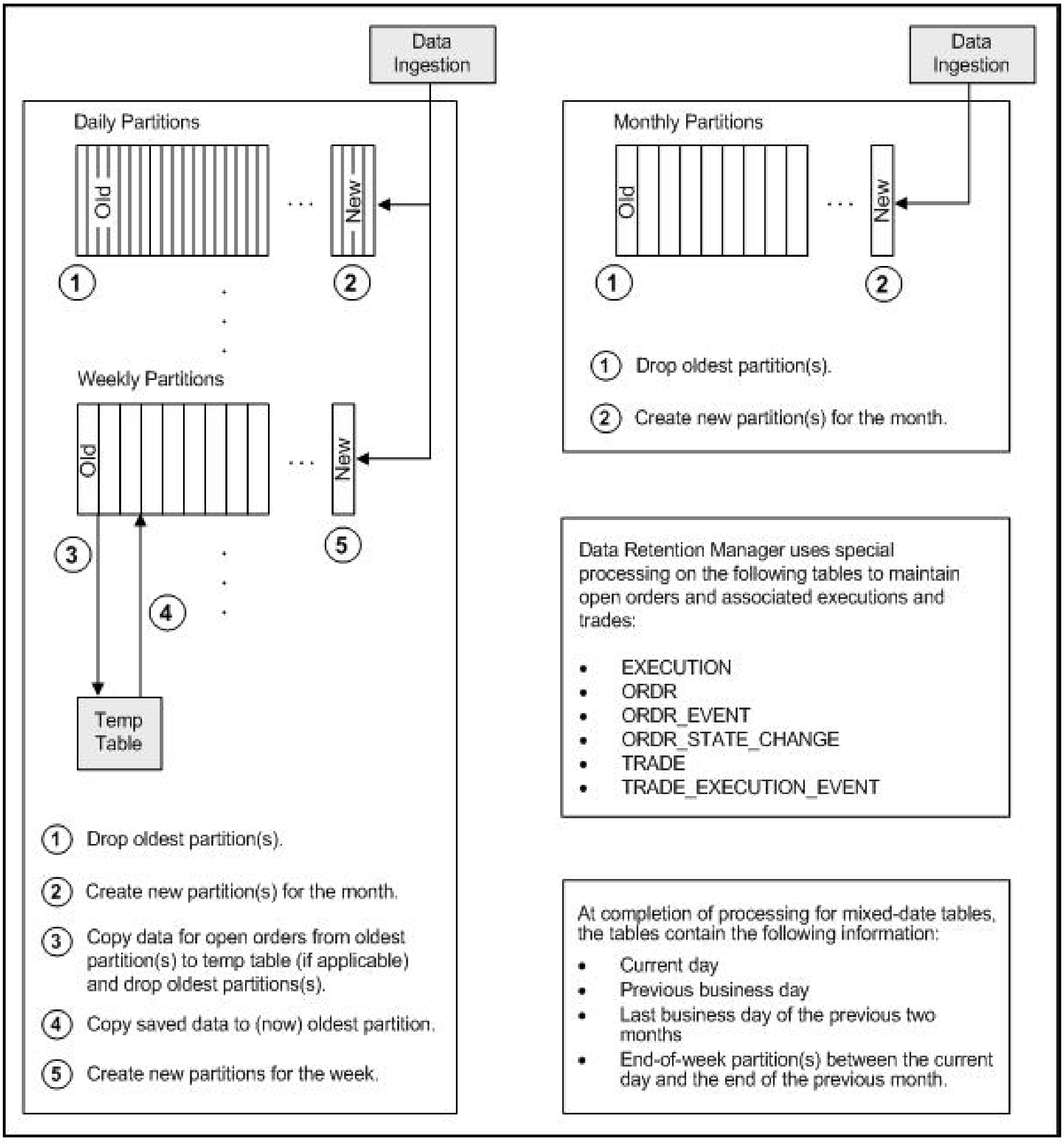

Data Ingestion
Data Ingestion
Daily Partitions
Old
New
Weekly Partitions
Old
New
Temp Table
Monthly Partitions
Old
New
1 Drop oldest partition(s).
2 Create new partition(s) for the month.
3 Copy data for open orders from oldest partition(s) to temp table (if applicable) and drop oldest partitions(s).
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 26. 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 (Refer to Figure 16 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 27. Data Retention Manager Configuration

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

Table 46. 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 Financial Services 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 Financial Services 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 (Refer toFigure 27, for more information).

To Run the Data Retention Manager

To run 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_object_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 47

Table 47. 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 Financial Services 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 (Refer to Table 46).

This procedure eliminates the need to perform frequent index maintenance; Oracle Financial Services 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, Refer to Table 46 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, Refer to *Calendar Manager Utility,* on page 117, 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 Financial Services 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 Financial Services 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.

To Add a Monthly Database Partition

To add a monthly partition, run the utility's shell script as follows (Refer to 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.

To Drop a Monthly Database Partition

To drop a monthly partition, run the utility's shell script as follows (Refer to 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 48 describes the table's contents.

Table 48. 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.

Table 48. BUSINESS.KDD_DR_MAINT_OPRTN Table Contents

`PARTN_TYPE_CD`	Indicates the partition type: ● D: Daily ● W: Weekly ● M: Monthly ● X: Mixed Date
`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 49 provides the table's contents.

Table 49. 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 50 describes the table's contents.

Table 50. 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.
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. Refer to Table 44 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 about Password Manager Utility.

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 needs to 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 following are the users for which the script prompts for passwords, 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 need to 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 needs to be regenerated by running `FIC_HOME/AM/create_at_war.sh`.

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 143, 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 51 outlines the logging levels that the Common Logging component supports.

Table 51. 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/manta
s_admin_tools_message_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.

Ingestion Manager

The following file is the message library for the Ingestion Manager:

```
<INSTALL_DIR>/ingestion_manager/config/message.dat
```

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`
- Ingestion Manager:
  `<INSTALL_DIR>/ingestion_manager/config/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 in the below 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.
log.max.size=2000000000
```

Figure 28. 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 52 identifies the valid values for this property.

Table 52. 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>/<file-name>`	Records the logs to a file with the filename <filename>, which resides at <path>. For example, `log.message.library=/user/jsmith/message/mes-sages.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 53 identifies the configurable properties for logging in an Oracle Financial Services client's environment.

Table 53. Configurable Parameters for Common Logging

Property	Sample Value	Description
`log.format`	%d [%t] %p %m%n	Identifies the log formatting string. Refer to Apache Software's *Short Introduction to log4j* guide (http://log-ging.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 (Refer to *Log File Sizes,* on page 143, for more information).
`log.category.<catgory_name>.location`		Contains routing information for message libraries for this category. For more information (Refer to *Logging Location Property Values,* on page 143, 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 (Refer to *Logging Location Property Values,* on page 143, 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 53. 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.

The Ingestion Manager 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 52 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 Refer to the appropriate `readme.txt` file for information about updated IDs.